IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

INDUSTRIAL PRINT TECHNOLOGIES, LLC, Plaintiff v.	The Honorable Barbara M.G. Lynn
CENVEO, INC. and HEWLETT-PACKARD COMPANY	CASE NO. 3:15-CV-00165-M
O'NEIL DATA SYSTEMS, INC. and HEWLETT-PACKARD COMPANY	CASE NO. 3:15-CV-01100-M
O'NEIL DATA SYSTEMS, INC. and HEWLETT-PACKARD COMPANY	CASE NO. 3:15-CV-01101-M
QUAD/GRAPHICS, INC. and HEWLETT-PACKARD COMPANY	CASE NO. 3:15-CV-01103-M
O'NEIL DATA SYSTEMS, INC. and HEWLETT-PACKARD COMPANY	CASE NO. 3:15-CV-01104-M
VISTAPRINT U.S.A., INC. and HEWLETT-PACKARD COMPANY	CASE NO. 3:15-CV-01106-M
FORT DEARBORN COMPANY and HEWLETT-PACKARD COMPANY, Defendants.	CASE NO. 3:15-CV-01195-M

IPT'S APPENDIX TO ITS OPPOSITION TO DEFENDANTS HP INC., O'NEIL DATA SYSTEMS INC., CENVEO, INC. AND FORT DEARBORN COMPANY'S MOTION TO DECLARE THIS AN EXCEPTIONAL CASE FOR ATTORNEYS' FEES AND COSTS

PUBLIC REDACTED VERSION

Date: January 12, 2017

/s/ Timothy P. Maloney

Timothy P. Maloney (IL 6216483)
Nicole L. Little (IL 6297047)
David A. Gosse (IL 6299892)
FITCH, EVEN, TABIN & FLANNERY LLP
120 South LaSalle Street, Suite 1600
Chicago, Illinois 60603
Telephone: (312) 577-7000
Facsimile: (312) 577-7007

Attorneys for Plaintiff

APPENDIX INDEX

Exhibits to Gauthier Declaration

Exhibits to Raasch Declaration

Exhibits to Maloney Declaration

Additional Appendix Materials

Exhibit 1
to Gauthier Declaration

PODi

January 2011

Personalized Print Markup Language

Application Notes

PODi: the Digital Printing Initiative

1240 Jefferson Road, Rochester, New York 14623, USA

Tel: (585) 239-6014

Internet: http://www.podi.org

January 2011
PPML Application Notes

PODi the Digital Printing Initiative

Approval of a PODi standard requires acceptance by the members of PODi.

PODi is a not for profit industry consortium formed in 1996. Its charter is to foster the growth of the digital printing industry through market and standards development activities. PODi constantly monitors market and technology trends in the industry, and shares information through seminars, independent research, white papers, articles, and the web. PODi promotes interoperability through the PPML suite of open, XML based standards, test suites and certification.

PODi welcomes feedback on this document. Please send comments via email to ppmlinfo@podi.org.

Contents

Foreword

The Personalized Print Markup Language (PPML) standard was introduced in May 2000 by PODi to foster market growth in high-volume, full-color variable data printing. There are several documents that are a part of the PPML library. The purpose of this document is to aid developers of Consumer and Producer solutions.

These application notes were developed by the Technical Working Group during the development of the PPML specification.

NOTE Some of the elements of the PPML standard may be the subject of patent rights. PODi is not responsible for identifying any or all such patent rights.

PODi does not guarantee the suitability of PPML or any of the conformance subsets for any specific purpose.

PODi Senior Technologist: Dr. Paul Jones

PODi Director of Technology: James Mekis

Send suggestions for improving this document to PODi, 1240 Jefferson Road, Rochester, NY 14623, USA; e-mail: ppmlinfo@podi.org.

Personalized Print Markup Language Application Notes

Introduction

The Personalized Print Markup Language (PPML) specification defines an XML grammar for specifying graphical page content for both monochrome and full color variable data jobs. The PPML format describes how to combine existing digital assets using clipping and transformations into pages, documents, and sets. PPML provides meta information that can be used to guide PPML-based workflows. PODi recommends the use of JDF to describe such workflows. For information on the use of PPML with JDF, see the PODi *Digital Print Ticket (DPT) Specification*.

The purpose of PPML is to:

- optimize ripping and print speed,
- organize page content for flexible processing and finishing,
- allow flexible access to digital assets, which may be stored internally, locally, or remotely,
- leverage existing standards, infrastructures and digital assets,
- enable interoperability.

The Application Notes for the Personalized Print Markup Language (PPML) are intended to provide working examples to demonstrate proper PPML coding to achieve desired formatting. They are a companion to, but not a replacement for, the PPML library of functional and conformance specifications.

This document discusses the PPML imaging model and how that model impacts the rendering of objects. Examples of PPML code are included with explanations of how these examples are rendered as the result of transforms, clipping, and positioning. These guidelines should be used to guide producer and consumer implementations.

The sections on Supplied Resources and Reusable Content show example code for identifying and reusing resources and content. Examples show both proper and improper code to help developers focus on best practices.

Also included are sections on PostScript to PPML and PDF to PPML conversion to identify packaging differences for resources. Code examples are provided.

Imaging model

PPML defines how graphical elements are composed into pages, documents and jobs. The imaging model defines the final appearance when multiple graphical elements are placed onto a medium. It specifically defines the appearance of overlapping graphical elements.

In PPML, each graphical element is represented by a PPML object. A page in PPML is defined as a sequence of PPML objects, where each object will be rendered in the order they are defined. The imaging model defines how an object is combined into the content constructed from the objects preceding it in the PPML page definition.

Every PPML object has a binary mask associated with it that defines which areas are transparent and which are opaque. Conceptually, that mask defines the area erased from the page before the object is rendered onto that page. The binary mask also restricts the rendering area for that object.

The background of the content data that defines the object is initially considered transparent. Drawing commands in the content data update the binary mask to erase the background.

In image data, the alpha channel or image mask defines the binary mask for the object. The drawing areas of partially transparent colors set those areas in the mask to opaque. No mixing of colors will occur for partially transparent objects that overlap.

An **alpha channel** is a type of channel used in graphics software for saving selections. Most bitmap editing software allows you to save multiple alpha channels with an image when it is saved in the program's native file format. Any of the alpha channels can be reloaded as a selection or mask at any time, even after closing and reopening the image.

A few of the standard image formats (TIFF and PNG, for example) provide support for an embedded alpha channel which represents up to 256 levels of transparency. Images with an embedded alpha channel can be ported to other applications while retaining transparency as long as the other application also supports alpha channels. Like a mask, the darkest areas of an alpha channel is most transparent, white areas are opaque, and shades of gray represent varying levels of transparency.

http://graphicssoft.about.com

Note that PPML/GA does not allow the use of image data with alpha channels.

The binary mask is defined to be transparent outside the area specified by the dimensions of that object. The dimensions of an object are specified by the *Dimensions* attribute of the **SOURCE** element in the definition of that object.

The binary mask is further modified by any clipping specified for that object. Clipping is introduced by the *ClippingBox* attribute on the **SOURCE** element.

In PPML, an object can be placed onto a page using several steps of transformation and clipping. Each **VIEW** element defines one step of transformation and clipping. The **TRANSFORM** sub-element in the **VIEW** element defines a transformation of the object coordinate system.

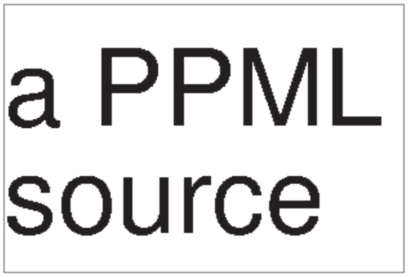
Transformations are specified by a matrix that has the same syntax and semantics as transform matrices defined in the PostScript and PDF specifications. After transformation, any clipping defined by a **CLIP_RECT** sub-element is applied. Clipping and transformation apply to both the object itself and its binary mask.

Notes on Transforming, Clipping and Positioning

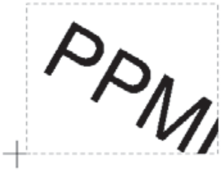
The following two examples show how to process a simple case of a **MARK** on a PPML page: a single EPS file is transformed and clipped in various ways and then placed on a page. All of the instructions in the first example will be contained in the **MARK** element; the second example shows how the same result could be accomplished using a **REUSABLE_OBJECT** element.

Both examples use the same original EPS file – a few words of text, which fits into a box that is 100 units high and 150 units wide. The result we want to achieve is a part of this EPS file, reduced, cropped, and rotated, as shown at the right.

Source



Desired Result



Self-Contained MARK Example

A self-contained **MARK** has this structure:

The simplest possible **MARK** contains one **VIEW** element and one **OBJECT** element.

- An **OBJECT** is a **VIEW** of a single **SOURCE**.
- Each of the **VIEW**s can contain a **TRANSFORM** and a **CLIP_RECT**.

To process a **MARK**, the Consumer must first process each **OBJECT** inside it. To do that, it first processes the **SOURCE** in the **OBJECT**. Here is the sequence the Consumer must follow:

- Process the **SOURCE**, applying its *ClippingBox*, if any**.**
- Take the result and transform it using the **TRANSFORM** from the **OBJECT**'s **VIEW.**
- Take the result and clip it using the **CLIP_RECT** from the **OBJECT**'s **VIEW.**

This produces one **OBJECT** that will be contained in the **MARK**.

Now, position the **OBJECT** in the **MARK**'s coordinate space.

Repeat the above steps for each **OBJECT** in the **MARK**.

Now, apply the **MARK**'s **VIEW**:

- Take the set of **OBJECT**s (one or more) and transform it using the **TRANSFORM** from the **MARK**'s **VIEW.**

- Take the result and clip it using the **CLIP_RECT** from the **MARK**’s **VIEW**.

This produces the final piece of page content that will appear on the page. The last step will be to position it on the page, using the **MARK**’s *Position* attribute.

The following PPML fragment achieves our desired result using a self-contained **MARK**:

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Format=”application/postscript”
                Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

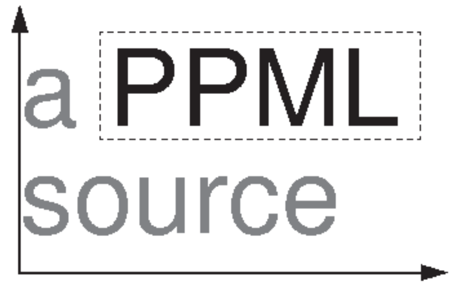
A PPML Consumer processes this fragment using the following steps:

1. Read the SOURCE element in the OBJECT

First, the Consumer finds the **SOURCE** element inside the **MARK**:

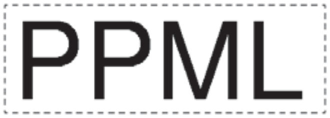
```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

The *ClippingBox* attribute crops the edges of the EPS file, as shown by the dashed line:



☑ *Current coordinate space: the* **SOURCE**.

This is the content defined by this SOURCE element:



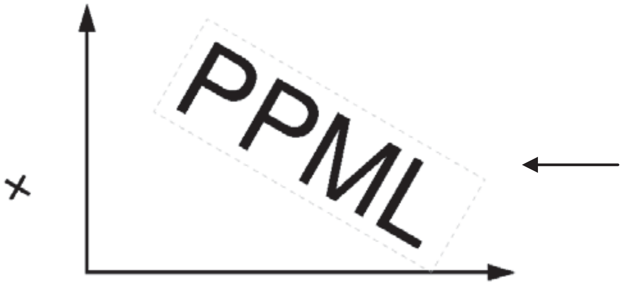
+ SOURCE's origin

2. Completing the OBJECT: Apply VIEW

Next, the Consumer applies the **OBJECT**'s **VIEW**, starting with the **TRANSFORM** element:

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

The transformation component of this **VIEW** specifies a translation of (-25.98,31.7) and a rotation of –30°.



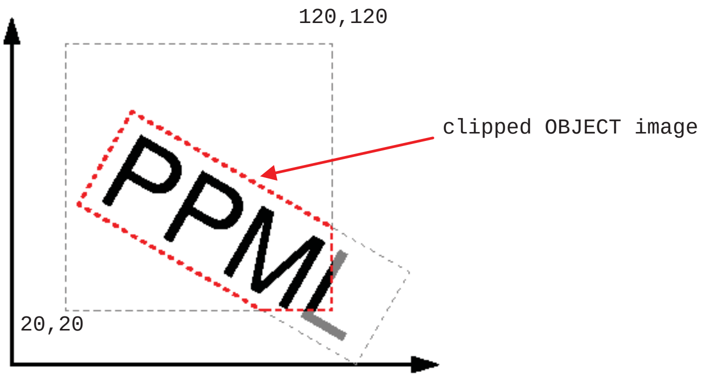
SOURCE origin
Offset -25.98,31.7
from OBJECT origin,
rotated -30°

☑ *Current coordinate space: the* **OBJECT**.

Now process the **OBJECT**'s **CLIP_RECT**.

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECTPosition="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

The **CLIP_RECT** (20,20 to 120,120) clips the rotated image like this:

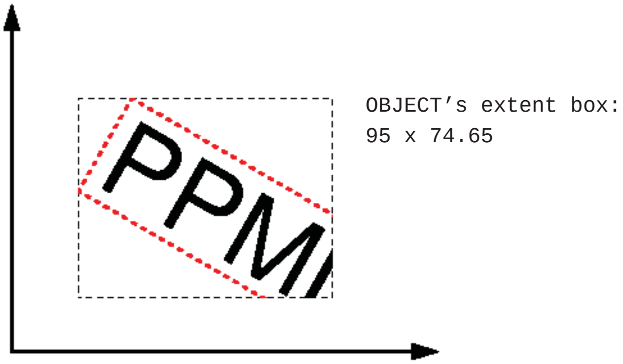


☑ *Current coordinate space: the* **OBJECT**.

Note

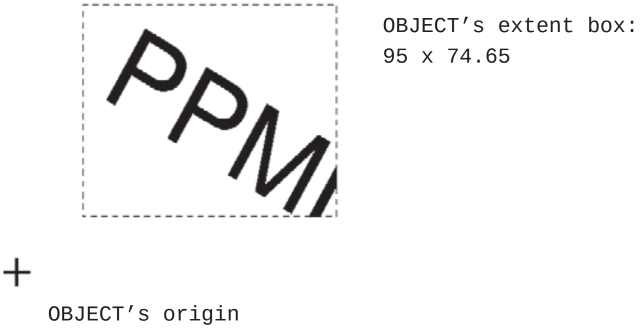
The drawings use color to highlight the clipping area.

Next, determine the extent box of this **OBJECT** element:



☑ *Current coordinate space: the* ***OBJECT****.*

This is the content that this **OBJECT** element defines:

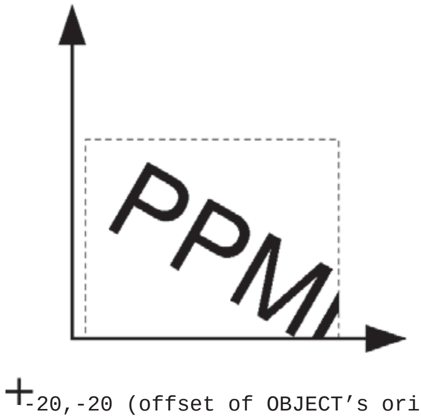


3. Place the OBJECT in the MARK, and apply the MARK’s VIEW

A **MARK** can contain several **OBJECT**s, each with its own position. When each **OBJECT** is complete, its origin can be placed anywhere within the coordinates of its enclosing **MARK** element. This is done using the **OBJECT** element’s *Position* attribute.

In this example the **MARK** contains only one **OBJECT**, positioned at (-20,-20).

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```



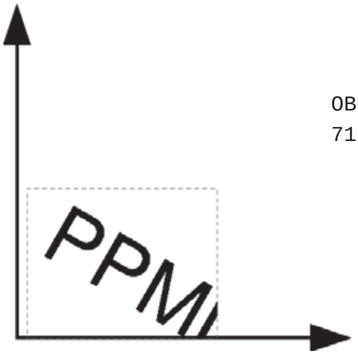
-20,-20 (offset of OBJECT’s origin in the MARK)

☑ *Current coordinate space: the **MARK**.*

Next, apply the **MARK**'s **TRANSFORM**: scale the **OBJECT** to 75% of its original size:

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

Result:



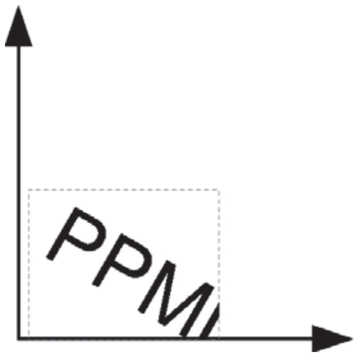
OBJECT's extent box in the MARK:
71.25 x 56

☑ *Current coordinate space: the **MARK**.*

Next, apply the **MARK***'s* **CLIP_RECT***:* in this case, the extra clipping does not influence the result.

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

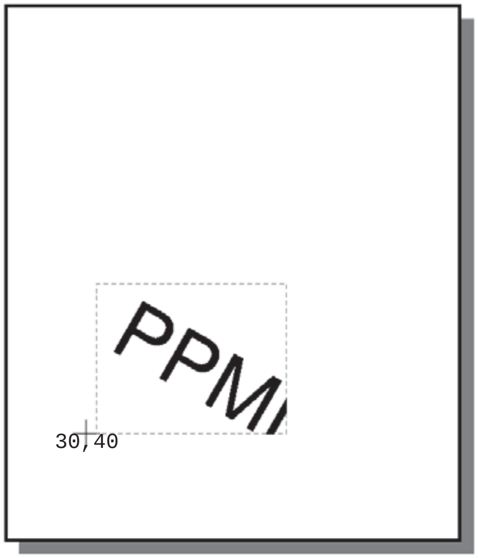
The **MARK**’s content is now complete. The content can now be positioned on the page, as shown below.



4. Position the MARK on the page.

The only remaining step is to process the **MARK** element’s *Position* attribute.

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```



☑ Current coordinate space: the PAGE.

The entire **MARK** is now complete: the content has been marked onto the page.

```
<MARK Position="30 40">
  <VIEW>
    <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
    <CLIP_RECT Rectangle="0 0 75 75" />
  </VIEW>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
</MARK>
```

The following PostScript code could be placed before the EPS source to produce this result:

```
30 40 translate                                % MARK position
0 0 75 75 rectclip                             % MARK clipping
[0.75 0 0 0.75 0 0] concat                     % MARK transform
-20 -20 translate                              % OBJECT position
20.0 20.0 100.0 100.0 rectclip                 % OBJECT clipping
[0.866 -0.5 0.5 0.866 -25.98 31.7] concat      % OBJECT transform
30.0 50.0 120.0 40.0 rectclip                  % SOURCE clipping
% insert content of file "ppml.eps" here
```

REUSABLE_OBJECT Example

This example renders the same content as the previous example, but uses a **REUSABLE_OBJECT**.

A **REUSABLE_OBJECT** has this structure:

- The simplest possible **REUSABLE_OBJECT** contains a **VIEW**, one **OBJECT**, and an **OCCURRENCE_LIST** with one **OCCURRENCE**.
- Each **OCCURRENCE** specifies a **VIEW** of all the ***OBJECT***s in this **REUSABLE_OBJECT**.
- A **MARK** can include a particular **OCCURRENCE** of a **REUSABLE_OBJECT** by including an **OCCURRENCE_REF**.
- It only makes sense to use **REUSABLE_OBJECT** if its **OCCURRENCE**s are used in more than one **MARK**; it is probable (but not required) that the PPML Consumer will optimize the **OBJECT** for reuse.

To process a **REUSABLE_OBJECT**, the Consumer must first process each **OBJECT** inside it. To do that, it first processes the **SOURCE** in the **OBJECT**. It is the same sequence as is used for **OBJECT**s within a **MARK**:

- Process the **SOURCE**, applying its *ClippingBox* if any.
- Take the result and transform it using the **TRANSFORM** from the **OBJECT**'s **VIEW.**
- Take the result and clip it using the **CLIP_RECT** from the **OBJECT**'s **VIEW.**

This produces one **OBJECT** that will be contained in the **REUSABLE_OBJECT**.

Now, position the **OBJECT** in the **REUSABLE_OBJECT**'s coordinate space.

Repeat the above for each **OBJECT** in the **REUSABLE_OBJECT**.

Now, apply the **REUSABLE_OBJECT**'s **VIEW**:

- Take the set of (one or more) **OBJECT**s and transform it using the **TRANSFORM** from the **REUSABLE_OBJECT**'s **VIEW.**
- Take the result and clip it using the **CLIP_RECT** from the **REUSABLE_OBJECT**'s **VIEW.**

Now, apply each **OCCURRENCE**'s **VIEW**:

- Take the result and transform it using the **TRANSFORM** from the **OCCURRENCE**'s **VIEW .**
- Take the result and clip it using the **CLIP_RECT** from the **OCCURRENCE**'s **VIEW .**
- Repeat the above for each **OCCURRENCE** in the **OCCURRENCE _LIST.**

This process produces the final piece of page content for each **OCCURRENCE**. They are now ready to be included on a page with an **OCCURRENCE_REF**. The last step will be to position the content on the page, using the **MARK**'s *Position* attribute.

The following PPML fragment achieves our desired result using a **REUSABLE_OBJECT**:

```
<REUSABLE_OBJECT>
   <OBJECT Position="-20 -20">
      <SOURCE Format="application/postscript"
               Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
  <VIEW />
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="example">
         <VIEW>
            <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
            <CLIP_RECT Rectangle="0 0 75 75" />
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>

<MARK Position="30 40">
   <OCCURRENCE_REF Ref="example" />
</MARK>
```

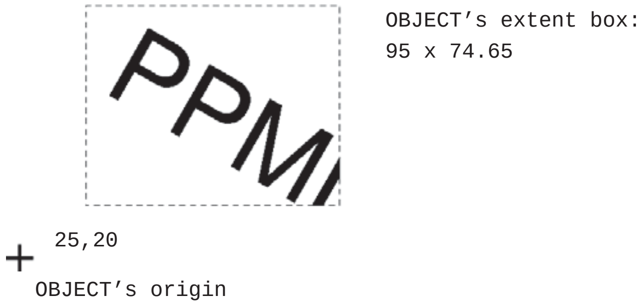
A PPML Consumer processes this fragment using the following steps.

1. Create the OBJECT specified in the REUSABLE_OBJECT.

Use steps 1 and 2 from the previous example to obtain the **OBJECT** by reading its **SOURCE** and applying its **VIEW**.

```
<REUSABLE_OBJECT>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
  <VIEW />
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="example">
         <VIEW>
            <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
            <CLIP_RECT Rectangle="0 0 75 75" />
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

This is the content that this OBJECT element defines**:**



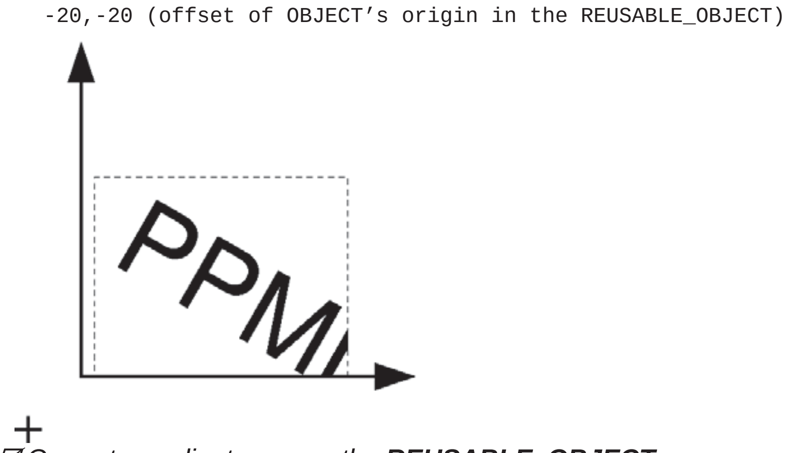
☑ *Current coordinate space: the* **OBJECT**

2. Place the OBJECT, and apply the REUSABLE_OBJECT’s and OCCURRENCE’s VIEWs.

A **REUSABLE_OBJECT** can contain several **OBJECT**s, each with its own position. Thus, when each **OBJECT** is complete, its origin can be placed anywhere within the coordinates of its enclosing **REUSABLE_OBJECT** element. This is done using the **OBJECT** element’s *Position* attribute.

In this example, the **OBJECT** is positioned at (-20,-20).

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```



☑ *Current coordinate space: the* ***REUSABLE_OBJECT****.*

Next, apply the **REUSABLE_OBJECT**’s **VIEW**: transform and clip the **OBJECT** as specified. In this example, the **REUSABLE_OBJECT**’s **VIEW** is empty and no processing is required.

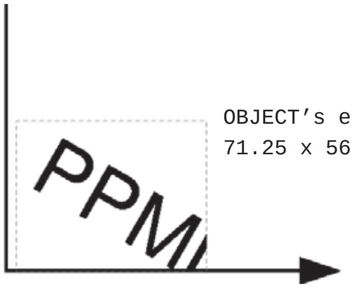
```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
 <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

Next, apply the **OCCURRENCE**'s **TRANSFORM**: scale the **OBJECT** to 75% of its current size:

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
 <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
```

```
         <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

Result:



OBJECT’s extent box in the OCCURRENCE:
71.25 x 56

☑ *Current coordinate space: the* **OCCURRENCE**.

Next, apply the **OCCURRENCE**’s **CLIP_RECT**: in this case, the extra clipping has no effect.

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
 <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

The **OCCURRENCE**’s content is now complete**.**

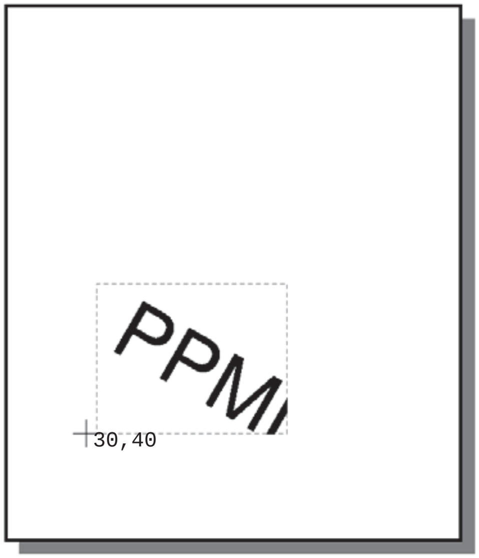
```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

3. Position the OCCURRENCE on the PAGE.

The only remaining step is to apply the **MARK** element’s *Position* attribute to the **OCCURRENCE** created in step 2:

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

```
<MARK Position="30 40">
  <OCCURRENCE_REF Ref="example" />
<MARK/>
```



☑ Current coordinate space: the PAGE.

The entire **MARK** is now complete: the content has been marked onto the page.

Supplied resources

When using PostScript content data in a PPML dataset, each PostScript file typically contains a similar prologue and uses the same set of fonts. To reduce the size and processing time of those PostScript files, PPML includes the ability to allow fonts and PostScript ProcSets to be specified once such that they may be used throughout the PPML dataset.

Typically the prologue of a PostScript file can be converted into a PostScript ProcSet creating PostScript “snippets” that can be used throughout a PPML dataset without having to include that PostScript prologue in each of the PostScript “snippets”. Similarly, fonts supplied in PPML need not be included in each PostScript “snippet” that uses those fonts.

PPML uses the **SUPPLIED_RESOURCE** element to define a font or PostScript ProcSet for reuse. The **SUPPLIED_RESOURCE_REF** element is used to define which content data may depend on a supplied resource. The **SUPPLIED_RESOURCE_REF** element ensures that the supplied resource is in memory for those pieces of content data requiring that supplied resource. If a supplied resource is not a required resource for a given piece of content data that supplied resource may not be referenced in that content data.

PPML uses the **SUPPLIED_RESOURCE** element to define a font or PostScript ProcSet for reuse.

A PPML Consumer may optimize the use of supplied resources by making the supplied resource persistent at the point of definition. A Consumer should be prepared to handle multiple resources with the same name (defined in different PPML scopes). A PPML Consumer must ensure that a supplied resource is available when the supplied resource is a required resource for a piece of content data. Note that a PPML Consumer is not required to disable access to a supplied resource for content data that do not require that supplied resource. A PPML Consumer must however ensure that the correct version of a supplied resource is available to content data. Supplied resources take precedence over any resource installed in a PPML Consumer.

In a level 1 PPML/GA dataset no content data may depend on resources installed in a PPML Consumer, therefore all resources needed must be supplied in that PPML dataset. In a level 2 PPML/GA dataset references to resources installed in a PPML Consumer are possible. In such cases, that dataset may not print correctly if processed by a different PPML Consumer.

PPML supports the definition of supplied resources which depend on other supplied resources. The required resources at the point of definition of a supplied resource must be in memory before that supplied resource is loaded into memory. When that supplied resource is referenced, those required resources may no longer be required. It is up to the PPML Producer to ensure that at the point of reference of a supplied resource the resources that still need to be in memory are required resources for the content data referencing that supplied resource.

Example:

```
ProcSet X:
   1 dict begin
      /myfunc { /Y /ProcSet findresource /dosomething get exec } def
   currentdict end
   /X exch /ProcSet defineresource
ProcSet Y:
   1 dict begin
      /dosomething { (hello) show } def
   currentdict end
   /Y exch /ProcSet defineresource
ProcSet Z:
   1 dict begin
      /myfunc /Y /ProcSet findresource /dosomething get def
   currentdict end
   /Z exch /ProcSet defineresource
```

In this example, X is dependent on Y when X is used. Y need not be in memory when X is loaded into memory. Z however is only dependent on Y when Z is loaded into memory. Y need not be in memory when Z is used. Therefore, Y must be required resource at the point of definition of Z and Y must be a required resource whenever X is a required resource.

For Y to be a required resource for Z, Y must be defined in a different scope from Z and made a required resource for the scope in which Z is defined. Y and Z cannot be defined in the same scope, as required resources defined in a scope are *not* required resources for supplied resource defined in the same scope. Therefore, the following example will *not* work for Y and Z but will work for Y and X:

```
   <PAGE>
   <SUPPLIED_RESOURCES>
      <SUPPLIED_RESOURCE Type=”ProcSet” Name=”Y” ResourceName=”Y”
Format=”application/postscript”>
         <EXTERNAL_DATA Src=”Y.ps”/>
      </SUPPLIED_RESOURCE>
      <SUPPLIED_RESOURCE Type=”ProcSet” Name=”Z” ResourceName=”Z”
Format=”application/postscript”>
         <EXTERNAL_DATA Src=”Z.ps”/>
      </SUPPLIED_RESOURCE>
   </SUPPLIED_RESOURCES>
   <REQUIRED_RESOURCES>
      <SUPPLIED_RESOURCE_REF Ref=”Y”/>
      <SUPPLIED_RESOURCE_REF Ref=”Z”/>
   </REQUIRED_RESOURCES>
   <MARK Position=”0 0”>
```

```
          :
          reference Z
          :
      <MARK>
```

Some developers try to streamline the process with the following solution, ***which is not supported***:

```
      <PAGE>
      <SUPPLIED_RESOURCES>
          <SUPPLIED_RESOURCE Type=”ProcSet” Name=”Y” ResourceName=”Y”
  Format=”application/postscript”>
              <EXTERNAL_DATA Src=”Y.ps”/>
          </SUPPLIED_RESOURCE>
      </SUPPLIED_RESOURCES>
      <REQUIRED_RESOURCES>
          <SUPPLIED_RESOURCE_REF Ref=”Y”/>
      </REQUIRED_RESOURCES>
      <SUPPLIED_RESOURCES>
          <SUPPLIED_RESOURCE Type=”ProcSet” Name=”Z” ResourceName=”Z”
  Format=”application/postscript”>
              <EXTERNAL_DATA Src=”Z.ps”/>
          </SUPPLIED_RESOURCE>
      </SUPPLIED_RESOURCES>
      <REQUIRED_RESOURCES>
          <SUPPLIED_RESOURCE_REF Ref=”Z”/>
      </REQUIRED_RESOURCES>
      <MARK Position=”0 0”>
          :
          reference Z
          :
      <MARK>
      </PAGE>
```

The above example shows invalid PPML as the PPML syntax rules do not allow multiple **SUPPLIED_RESOURCES** and **REQUIRED_RESOURCES** elements within a single scope. The correct method for creating a dependency of Z on Y therefore is:

```
      <DOCUMENT>
          <SUPPLIED_RESOURCES>
          <SUPPLIED_RESOURCE Type=”ProcSet” Name=”Y” ResourceName=”Y”
  Format=”application/postscript”>
```

```
            <EXTERNAL_DATA Src=”Y.ps”/>
         </SUPPLIED_RESOURCE>
      </SUPPLIED_RESOURCES>
      <REQUIRED_RESOURCES>
         <SUPPLIED_RESOURCE_REF Ref=”Y”/>
      </REQUIRED_RESOURCES>
      <PAGE>
      <SUPPLIED_RESOURCES>
         <SUPPLIED_RESOURCE Type=”ProcSet” Name=”Z” ResourceName=”Z”
            Format=”application/postscript”>
            <EXTERNAL_DATA Src=”Z.ps”/>
         </SUPPLIED_RESOURCE>
      </SUPPLIED_RESOURCES>
      <REQUIRED_RESOURCES>
         <SUPPLIED_RESOURCE_REF Ref=”Z”/>
      </REQUIRED_RESOURCES>
      <MARK Position=”0 0”>
         :
         reference Z
         :
      <MARK>
      </PAGE>
   </DOCUMENT>
```

Reusable content

In many variable data print (VDP) jobs certain objects are used multiple times (such as logos, signatures and page backgrounds). These objects can be considered *reusable*. Establishing objects as reusable may enable the Consumer to optimize the rendering of those objects on a page. Examples of such optimizations include pre-rasterization, pre-ripping and caching.

Reusable object definitions are *scoped*. Scoping a reusable object restricts references to that reusable object to a well-defined part of the PPML hierarchy. Scoping informs the Consumer when a reusable object can no longer be referenced allowing the definition of that reusable object to be discarded.

To support PPML Producers that generate PPML in a stream, PPML allows *scope promotion*. Scope promotion allows the definition of a reusable object to extend beyond the scope in which the definition occurs. Scope promotion is useful if a Producer does not know in advance if a reusable object will be needed or when that reusable object will be used. If the Producer detects the need for a reusable object, its definition can only be made in the scope currently being defined in the PPML stream. If the Producer expects that the reusable object will be needed after the end of the scope being defined, scope promotion may be used to avoid additional definitions of that reusable object.

The concept of an *occurrence* was introduced to inform a Consumer of how a reusable object will be placed on to a page. Knowing the rotation and scaling of the reusable object up front simplifies the process of rendering a reusable object on a page in an optimized manner. By only allowing an occurrence to be positioned (without additional scaling or rotation) onto a page, pre-rasterization of the reusable object becomes a viable optimization method.

To avoid repeated definitions of frequently used reusable objects and resources in different PPML datasets the concept of global scope was introduced. Using scope promotion, an occurrence or resource can be defined in global scope. This allows an occurrence or resource definition to be referenced without having to provide that definition in PPML again. The submitter of a PPML dataset containing such references will have to ensure that the appropriate definitions are available to the Consumer.

The use of global definitions across multiple PPML datasets may introduce dataset interdependencies with respect to the order in which the Consumer must process those datasets. A Consumer is not required to keep global definitions indefinitely and may not have the ability to do so. The Producer must coordinate with the Consumer to ensure that such workflows are reliable. Global occurrences can be useful in facilitating a set of PPML datasets using consistent versions of those occurrences (such as a large image library), where the generator of those PPML datasets need not be able to recreate the definition of those occurrences.

A Consumer is not required to keep global definitions indefinitely and may not have the ability to do so.

A Producer must take care when redefining a global occurrence with very different content. It is recommended that a Producer issue new identifiers instead of re-using existing identifiers. Existing global definitions may be deleted by providing a new definition with the *Overwrite* attribute set to "*Delete*".

The concept of *environment* was introduced to avoid problems with similarly named global reusable object definitions by different customers. The environment is used to separate the global reusable object definitions based on a named context. Often a unique name, such as a registered domain name or the company name, is used to ensure that global reusable object definitions are not accidentally replaced. Producers may wish to use a GUID (*Global Unique Identifier*) or UUID (Universal Unique Identifier) to provide a unique name for each occurrence.

The following PPML dataset defines a reusable global image library of 100 images:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”2”/>
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”image/tiff” Dimensions=”72 72”>
                <EXTERNAL_DATA Src=”image001.tiff”/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name=”image001” Environment=”www.podi.org”
Scope=”Global”/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
            :
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”image/tiff” Dimensions=”72 72”>
                <EXTERNAL_DATA Src=”image100.tiff”/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name=”image100” Environment=”www.podi.org”
Scope=”Global”/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
</PPML>
```

Those images may be used in a separate PPML dataset as follows:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”2”/>
    <DOCUMENT_SET>
        <DOCUMENT>
            <PAGE>
                <MARK Position=”500 700”>
                    <OCCURRENCE_REF Ref=”image053”
                        Environment=”www.podi.org”/>
                </MARK>
```

```
             <MARK Position=”50 800”>
                 variable content defined here
             </MARK>
          </PAGE>
          <PAGE>
             <MARK Position=”500 700”>
                 <OCCURRENCE_REF Ref=”image075”
                     Environment=”www.podi.org”/>
             </MARK>
             <MARK Position=”50 800”>
                 variable content defined here
             </MARK>
          </PAGE>
       </DOCUMENT>
   </DOCUMENT_SET>
</PPML>
```

Deleting the Image Library

When the image library is no longer needed, the occurrences may be deleted by sending the following PPML dataset:

```
<PPML>
   <CONFORMANCE SubSet=”GA” Level=”2”/>
   <REUSABLE_OBJECT>
       <OBJECT Position=”0 0”>
          <SOURCE Format=”image/tiff” Dimensions=”72 72”>
             <INTERNAL_DATA/>
          </SOURCE>
       </OBJECT>
       <OCCURRENCES>
          <OCCURRENCE Name=”image001” Environment=”www.podi.org” Scope=”Global”
Overwrite=”Delete”/>
       </OCCURRENCES>
   </REUSABLE_OBJECT>
          :
   <REUSABLE_OBJECT>
       <OBJECT Position=”0 0”>
          <SOURCE Format=”image/tiff” Dimensions=”72 72”>
             <INTERNAL_DATA/>
          </SOURCE>
       </OBJECT>
       <OCCURRENCES>
          <OCCURRENCE Name=”image100” Environment=”www.podi.org” Scope=”Global”
Overwrite=”Delete”/>
       </OCCURRENCES>
   </REUSABLE_OBJECT>
```

```
</PPML>
```

Here, empty **INTERNAL_DATA** elements are used to define a reusable object with empty content. By setting the value of the *Overwrite* attribute on the **OCCURRENCE** element to “*Delete*”, the source data in the reusable object definition is ignored and the occurrence can no longer be referenced. Alternatively, it is also possible to supply the original reusable object definition and just set the *Overwrite* attribute to “*Delete*” for each **OCCURRENCE** element that is no longer needed.

It is also possible to include the above **REUSABLE_OBJECT** definitions into the last PPML dataset that uses those reusable objects. These **REUSABLE_OBJECT** definitions must be included at the end of the PPML dataset to ensure that the reusable objects are still available in the rest of the PPML dataset. The following example illustrates this:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”2”/>
    <DOCUMENT_SET>
        <DOCUMENT>
            <PAGE>
                <MARK Position=”500 700”>
                    <OCCURRENCE_REF Ref=”image053” Environment=”www.podi.org”/>
                </MARK>
                <MARK Position=”50 800”>
                    variable content defined here
                </MARK>
            </PAGE>
            <PAGE>
                <MARK Position=”500 700”>
                    <OCCURRENCE_REF Ref=”image075” Environment=”www.podi.org”/>
                </MARK>
                <MARK Position=”50 800”>
                    variable content defined here
                </MARK>
            </PAGE>
        </DOCUMENT>
    </DOCUMENT_SET>
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”image/tiff” Dimensions=”72 72”>
                <INTERNAL_DATA/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name=”image001” Environment=”www.podi.org”
```

```
Scope=”Global” Overwrite=”Delete”/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
            :
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”image/tiff” Dimensions=”72 72”>
                <INTERNAL_DATA/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name=”image100” Environment=”www.podi.org”
Scope=”Global” Overwrite=”Delete”/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
</PPML>
```

Dynamically positioning reusable objects

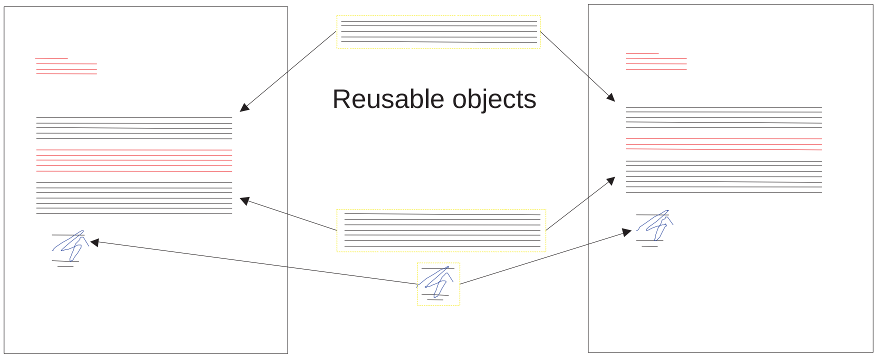
In many documents, certain parts of the document may be static but must be positioned dynamically to accommodate other objects that may have varying heights. For instance: a letter may have a paragraph that refers to the name and address of the recipient causing that paragraph to be 3 or 4 lines depending on the length of the name and address. Any paragraphs that follow must be moved up or down accordingly. If these paragraphs are completely static and the text of those paragraphs need not flow around other objects (such as images) those paragraphs can be optimized as a reusable object in PPML.

Example:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”1”/>
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”application/postscript”
                Dimensions=”75 400”>
                <EXTERNAL_DATA Src=”paragraph1.ps”/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name=”paragraph1”/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
```

```
<REUSABLE_OBJECT>
   <OBJECT Position=”0 0”>
      <SOURCE Format=”application/postscript”
         Dimensions=”100 400”>
         <EXTERNAL_DATA Src=”paragraph3.ps”/>
      </SOURCE>
   </OBJECT>
   <OCCURRENCES>
      <OCCURRENCE Name=”paragraph3”/>
   </OCCURRENCES>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position=”0 0”>
      <SOURCE Format=”image/tiff” Dimensions=”72 72”>
         <INTERNAL_DATA/>
      </SOURCE>
   </OBJECT>
   <OCCURRENCES>
      <OCCURRENCE Name=”signature”/>
   </OCCURRENCES>
</REUSABLE_OBJECT>
<DOCUMENT_SET>
   <DOCUMENT>
   <PAGE>
      <MARK Position=”30 700”>
         <OBJECT Position=”0 0”>
            <SOURCE Format=”application/postscript”
                Dimensions=”200 70”>
               <INTERNAL_DATA>
                  ... address block recipient 1...
               </INTERNAL_DATA>
            </SOURCE>
         </OBJECT>
      </MARK>
      <MARK Position=”30 500”>
         <OBJECT Position=”0 0”>
            <SOURCE Format=”application/postscript”
                Dimensions=”200 70”>
               <INTERNAL_DATA>
               ... greeting recipient 1 ...
            </INTERNAL_DATA>
         </SOURCE>
      </OBJECT>
   </MARK>
   <MARK Position=”30 488”>
      <OCCURRENCE_REF Ref=”paragraph1”/>
   </MARK>
   <MARK Position=”30 413”>
      <OBJECT Position=”0 0”>
         <SOURCE Format=”application/postscript”
             Dimensions=”400 35”>
            <INTERNAL_DATA>
               ... personalized paragraph ...
               ... 3 lines ...
            </INTERNAL_DATA>
         </SOURCE>
      </OBJECT>
   </MARK>
   <MARK Position=”30 388”>
      <OCCURRENCE_REF Ref=”paragraph3”/>
   </MARK>
   <MARK Position=”30 288”>
      <OCCURRENCE_REF Ref=”signature”/>
   </MARK>
</PAGE>
</DOCUMENT>
   :
<DOCUMENT>
<PAGE>
   <MARK Position=”30 700”>
      <OBJECT Position=”0 0”>
         <SOURCE Format=”application/postscript”
             Dimensions=”200 70”>
            <INTERNAL_DATA>
               ... address block recipient n...
            </INTERNAL_DATA>
         </SOURCE>
      </OBJECT>
```

```
          </MARK>
          <MARK Position=”30 500”>
             <OBJECT Position=”0 0”>
                 <SOURCE Format=”application/postscript”
                     Dimensions=”200 70”>
                    <INTERNAL_DATA>
                       ... greeting recipient n...
                    </INTERNAL_DATA>
                 </SOURCE>
             </OBJECT>
          </MARK>
          <MARK Position=”30 488”>
             <OCCURRENCE_REF Ref=”paragraph1”/>
          </MARK>
          <MARK Position=”30 413”>
             <OBJECT Position=”0 0”>
                 <SOURCE Format=”application/postscript”
                     Dimensions=”400 47”>
                    <INTERNAL_DATA>
                       ... personalized paragraph ...
                       ... 4 lines ...
                    </INTERNAL_DATA>
                 </SOURCE>
             </OBJECT>
          </MARK>
          <MARK Position=”30 376”>
             <OCCURRENCE_REF Ref=”paragraph3”/>
          </MARK>
          <MARK Position=”30 276”>
             <OCCURRENCE_REF Ref=”signature”/>
          </MARK>
       </PAGE>
       </DOCUMENT>
   </DOCUMENT_SET>
</PPML>
```



Using reusable objects for background imagery

In direct marketing applications, the variable data is often overlaid onto a full-color background image. Using reusable objects for background images can greatly improve RIPping and printing performance.

The following PPML shows how a reusable object may be used as a background in our previous example:

```
<PPML>
   <CONFORMANCE SubSet=”GA” Level=”1”/>
   <REUSABLE_OBJECT>
      <OBJECT Position=”0 0”>
         <SOURCE Format=”application/pdf”
            Dimensions=”595 842”>
            <EXTERNAL_DATA Src=”background.pdf”/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCES>
         <OCCURRENCE Name=”background”/>
      </OCCURRENCES>
   </REUSABLE_OBJECT>
   <REUSABLE_OBJECT>
      <OBJECT Position=”0 0”>
         <SOURCE Format=”application/postscript” Dimensions=”100 400”>
            <EXTERNAL_DATA Src=”paragraph3.ps”/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCES>
         <OCCURRENCE Name=”paragraph3”/>
      </OCCURRENCES>
   </REUSABLE_OBJECT>
   <REUSABLE_OBJECT>
```

```
    <OBJECT Position=”0 0”>
        <SOURCE Format=”image/tiff” Dimensions=”72 72”>
            <INTERNAL_DATA/>
        </SOURCE>
    </OBJECT>
    <OCCURRENCES>
        <OCCURRENCE Name=”signature”/>
    </OCCURRENCES>
</REUSABLE_OBJECT>
<DOCUMENT_SET>
    <DOCUMENT>
    <PAGE>
        <MARK Position=”0 0”>
            <OCCURRENCE_REF Ref=”background”/>
        </MARK>
        <MARK Position=”30 700”>
            <OBJECT Position=”0 0”>
                <SOURCE Format=”application/postscript”
                     Dimensions=”200 70”>
                    <INTERNAL_DATA>
                        ... address block recipient 1...
                    </INTERNAL_DATA>
                </SOURCE>
            </OBJECT>
        </MARK>
        <MARK Position=”30 500”>
            <OBJECT Position=”0 0”>
                <SOURCE Format=”application/postscript”
                     Dimensions=”200 70”>
                    <INTERNAL_DATA>
                        ... greeting recipient 1 ...
                    </INTERNAL_DATA>
                </SOURCE>
            </OBJECT>
        </MARK>
        <MARK Position=”30 488”>
            <OCCURRENCE_REF Ref=”paragraph1”/>
        </MARK>
        <MARK Position=”30 413”>
```

```
       <OBJECT Position=”0 0”>
          <SOURCE Format=”application/postscript”
              Dimensions=”400 35”>
             <INTERNAL_DATA>
                 ... personalized paragraph ...
                 ... 3 lines ...
             </INTERNAL_DATA>
          </SOURCE>
       </OBJECT>
   </MARK>
   <MARK Position=”30 388”>
       <OCCURRENCE_REF Ref=”paragraph3”/>
   </MARK>
   <MARK Position=”30 288”>
       <OCCURRENCE_REF Ref=”signature”/>
   </MARK>
</PAGE>
</DOCUMENT>
    :
<DOCUMENT>
<PAGE>
   <MARK Position=”0 0”>
       <OCCURRENCE_REF Ref=”background”/>
   </MARK>
   <MARK Position=”30 700”>
       <OBJECT Position=”0 0”>
          <SOURCE Format=”application/postscript”
              Dimensions=”200 70”>
             <INTERNAL_DATA>
                 ... address block recipient n...
             </INTERNAL_DATA>
          </SOURCE>
       </OBJECT>
   </MARK>
   <MARK Position=”30 500”>
       <OBJECT Position=”0 0”>
          <SOURCE Format=”application/postscript”
              Dimensions=”200 70”>
             <INTERNAL_DATA>
```

```
                    ... greeting recipient n...
                 </INTERNAL_DATA>
              </SOURCE>
           </OBJECT>
        </MARK>
        <MARK Position=”30 488”>
           <OCCURRENCE_REF Ref=”paragraph1”/>
        </MARK>
        <MARK Position=”30 413”>
           <OBJECT Position=”0 0”>
              <SOURCE Format=”application/postscript”
                  Dimensions=”400 47”>
                 <INTERNAL_DATA>
                    ... personalized paragraph ...
                    ... 4 lines ...
                 </INTERNAL_DATA>
              </SOURCE>
           </OBJECT>
        </MARK>
        <MARK Position=”30 376”>
           <OCCURRENCE_REF Ref=”paragraph3”/>
        </MARK>
        <MARK Position=”30 276”>
           <OCCURRENCE_REF Ref=”signature”/>
        </MARK>
     </PAGE>
     </DOCUMENT>
  </DOCUMENT_SET>
</PPML>
```

Note that here we add a reusable object before the other content placed on the page. Similarly, overlays such as “DRAFT” or “TEST” can be added using a reusable object placed after all the other content is placed on the page.

Global Supplied Resources

Like global reusable objects, Fonts and PostScript ProcSets can be downloaded into a PPML Consumer so that their definition may be omitted in other PPML datasets.

The following PPML dataset defines a global font:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”2”/>
    <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Name=”Lucinda-Grande”
            ResourceName=”LucindaGrande” Type=”Font” Scope=”Global”
            Environment=”www.podi.org” Format=”application/postscript”>
            <EXTERNAL_DATA Src=”lucindagrande.ps”/>
        </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
</PPML>
```

This global font may be referenced in other PPML datasets as follows:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”2”/>
    <REQUIRED_RESOURCES>
        <SUPPLIED_RESOURCE_REF Name=”Lucinda-Grande”
             Environment=”www.podi.org”/>
    </REQUIRED_RESOURCES>
    <DOCUMENT_SET>
        <PAGE>
            <MARK Position=”0 0”>
                <OBJECT Position=”0 0 “>
                    <SOURCE Format=”application/postscript”
                         Dimensions=”100 40”>
                        <INTERNAL_DATA>
                            (LucindaGrande) findfont 20 scalefont
                            0 5 moveto
                            (Hello world) show
                        </INTERNAL_DATA>
                    </SOURCE>
                </OBJECT>
            </MARK>
        </PAGE>
    </DOCUMENT_SET>
```

```
</PPML>
```

When the global font is no longer needed one can delete the global supplied resource by redefining the supplied resource with the *Overwrite* attribute set to “Delete”:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”2”/>
    <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Name=”Lucinda-Grande”
            ResourceName=”LucindaGrande” Type=”Font” Scope=”Global”
            Environment=”www.podi.org” Overwrite=”Delete”
            Format=”application/postscript”>
            <EXTERNAL_DATA Src=”lucindagrande.ps”/>
        </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
</PPML>
```

NOTE It is not necessary to provide the original definition when deleting a global supplied resource; an empty **INTERNAL_DATA** element may be used instead (see also **Deleting the Image Library** on Page 26).

Job ticketing

The use of **TICKET** and **TICKET_REF** for associating job ticket parameters with PPML pages is discouraged. Instead, a JDF job ticket should be used that refers to the PPML dataset as explained in the DPT 2.2 Application Notes.

The use of **TICKET_REF** with JDF is discouraged as **TICKET_REF** relies on the use of the resource update concept, which has been deprecated by CIP4 in JDF 1.3.

PostScript to PPML application notes

A typical PostScript file uses a prolog to define procedures used in the descriptions of the pages. In addition to the prolog some PostScript files embed font definitions. Some PostScript files also use PostScript forms for reusable content.

To convert such PostScript files to a PPML dataset, the procsets in the prolog, the embedded fonts, the forms and the page descriptions need to be packaged into separate entities.

A typical prolog will either create entries in `userdict` or will create separate dictionaries with those entries in them.

Example 1:

```
%%BeginProlog
/myfunc1 { ... } bind def
   :
/myfunc-n { ... } bind def
%%EndProlog
```

Example 2:

```
%%BeginProlog
/mydict 10 dict def
mydict begin
/myfunc1 { ... } bind def
   :
/myfunc-n { ... } bind def
end
%%EndProlog
```

In either case, a PostScript ProcSet should be defined that contains all the entries defined in the prolog. Such a ProcSet first creates a dictionary to contain all the entries; then that dictionary is defined as a named PostScript ProcSet using `defineresource`.

Example 1 can be transformed into a PostScript ProcSet named “myprocset” as follows:

```
<<
/myfunc1 { ... }
   :
/myfunc-n { ... }
>>
   (myprocset) exch /ProcSet defineresource pop
```

NOTE If multiple ProcSets are being used, their interdependencies must be made explicit in PPML. The order of definition of ProcSets in PPML may not be the same order in which ProcSets are loaded into VM (virtual memory). See also the notes on

Global Supplied Resources on Page 36.

Each page description must be packaged as a separate PostScript fragment that does not depend on the execution of other pages. The `showpage` operator need not be included.

Each of the page descriptions that used the functions defined in the original prolog will need some additional PostScript code to access the named ProcSet instead. The `findresource` operator may be used for this purpose as follows:

```
(myprocset) /ProcSet findresource begin
    : (insert original page description here)
end
```

Each embedded font definition will have to be packaged into a standalone font definition. This may include removing any references to functions defined in the prolog. Alternatively, the original embedded font definition can be made dependent on the named ProcSet created for the prolog as described for page descriptions. Note that the font will need to use the `findresource` operator to access the ProcSet.

For form definitions, the contents of the `PaintProc` procedure of the defined form must be packaged into a standalone PostScript fragment. This may include removing any references to functions defined in the prolog. Alternatively, the original embedded `PaintProc` definition can be made dependent on the named ProcSet created for the prolog as described for page descriptions.

Given the procset, standalone font, form and page descriptions, a first version of a PPML dataset can be created as follows (assuming A4 paper):

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”1”/>
    <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Type=”ProcSet” Format=”application/postscript”
            Name=”prolog” ResourceName=”myprocset”>
            <EXTERNAL_DATA Src=”myprocset.ps”/>
        </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
    <REQUIRED_RESOURCES>
        <SUPPLIED_RESOURCE_REF Ref=”prolog”/>
    </REQUIRED_RESOURCES>
    <PAGE_DESIGN TrimBox=”0 0 595 842”/>
    <JOB>
        <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Type=”Font”
            Format=”application/postscript”
            Name=”font-1” ResourceName=”fontname-1”>
            <EXTERNAL_DATA Src=”font-1.ps”/>
        </SUPPLIED_RESOURCE>
            :
            <SUPPLIED_RESOURCE Type=”Font”
            Format=”application/postscript”
            Name=”font-m” ResourceName=”fontname-m”>
            <EXTERNAL_DATA Src=”font-m.ps”/>
        </SUPPLIED_RESOURCE>
        </SUPPLIED_RESOURCES>
    <REQUIRED_RESOURCES>
        <SUPPLIED_RESOURCE_REF Ref=”font-1”/>
            :
        <SUPPLIED_RESOURCE_REF Ref=”font-m”/>
    </REQUIRED_RESOURCES>
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”application/postscript”
                Dimensions=”[form-1-urx] [form-1-ury]”
                ClippingBox=”[form-1-bounding-box]”>
                <EXTERNAL_DATA Src=”form-1.ps”/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCE_LIST>
            <OCCURRENCE Name=”form-1-view-1”>
                <VIEW>
                    <TRANSFORM Matrix=”[form-1-transform-1]”/>
                </VIEW>
            </OCCURRENCE>
                :
            <OCCURRENCE Name=”form-1-view-2”>
                <VIEW>
                    <TRANSFORM Matrix=”[form-1-transform-2]”/>
                </VIEW>
            </OCCURRENCE>
        </OCCURRENCE_LIST>
    </REUSABLE_OBJECT>
        :
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”application/postscript”
                Dimensions=”[form-k-urx] [form-k-ury]”
                ClippingBox="[form-k-bounding-box]">
                <EXTERNAL_DATA Src="form-k.ps"/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCE_LIST>
            <OCCURRENCE Name="form-k-view-1">
                <VIEW>
                    <TRANSFORM Matrix="[form-k-transform-1]"/>
                </VIEW>
            </OCCURRENCE>
                :
            <OCCURRENCE Name="form-k-view-2">
                <VIEW>
                    <TRANSFORM Matrix="[form-k-transform-2]"/>
                </VIEW>
            </OCCURRENCE>
        </OCCURRENCE_LIST>
    </REUSABLE_OBJECT>
    <DOCUMENT>
            <PAGE>
                <MARK Position="0 0">
                    <OBJECT Position="0 0">
                        <SOURCE Format="application/postscript"
                            Dimensions="595 842">
                            ClippingBox="[bbox-page-1-1]">
                            <EXTERNAL_DATA Src="page-1-1.ps"/>
                        </SOURCE>
                    </OBJECT>
                </MARK>
                <MARK Position="[x] [y]">
                    <OCCURRENCE_REF Ref="form-1-view-1"/>
                </MARK>
                <MARK Position="0 0">
                    <OBJECT Position="0 0">
                        <SOURCE Format="application/postscript"
                            Dimensions="595 842"
                            ClippingBox="[bbox-page-1-2]">
                            <EXTERNAL_DATA Src="page-1-2.ps"/>
                        </SOURCE>
                    </OBJECT>
                </MARK>
            </PAGE>
                :
            <PAGE>
                <MARK Position=”0 0”>
                    <OBJECT Position=”0 0”>
                        <SOURCE Format=”application/postscript”
                             Dimensions=”595 842”>
                            <EXTERNAL_DATA Src=”page-n.ps”/>
                        </SOURCE>
                    </OBJECT>
                </MARK>
            </PAGE>
        </DOCUMENT>
    </JOB>
    </PPML>
```

The above PPML dataset adheres to the PPML/GA specification, which is indicated by the inclusion of the **CONFORMANCE** element. To ensure that the font and form definitions have access to the procedures defined in the original prolog a **SUPPLIED_RESOURCE** for myprocset is created at the PPML level. The **SUPPLIED_RESOURCE_REF** in the **REQUIRED_RESOURCES** element at the PPML level is needed to inform the PPML Consumer that the myprocset ProcSet shall be made available in VM for all PostScript fragments used in the PPML scope.

The fonts, which may be dependent on the myprocset ProcSet, are defined at the **JOB** level such that they have a defined dependency on the myprocset ProcSet.

Each form is defined using a **REUSABLE_OBJECT** element. The bounding box of the original form definition is needed to inform the PPML Consumer of the area of the form coordinate system to cache. The *Dimensions* attribute introduces clipping to a bounding box of “0 0 w h” for *Dimensions=”w h”*. If the form's lower left corner has negative components, additional PostScript code must be added to translate the lower left corner of the form to (0,0). In that case, the bounding box for the form needs to be adjusted accordingly.

In PostScript, forms may depend on the graphics state from which the form is executed. This is not the case in PPML. Therefore, for such forms multiple **REUSABLE_OBJECT** definitions may be necessary. Those additional **REUSABLE_OBJECT** definitions would need to have the appropriate PostScript commands added to setup the graphics state in which the form is used.

To use a form, the page description will have to be split into two parts: the part before a form is executed and the part after a form is executed. The reference to the form will become an **OCCURRENCE_REF** in the PPML dataset. Each of the other parts is converted into a separate **MARK** element. The *ClippingBox* attribute of the **SOURCE** element should be added to reflect the area of the page that the part draws on.

Reusable content in PPML can only be translated when being placed onto a page. Therefore, for page descriptions that scale, rotate or shear a form, an **OCCURRENCE** must be defined with a **VIEW** element containing a **TRANSFORM** element that appropriately transforms the form. The form reference can then be implemented using a **MARK** element that contains an **OCCURRENCE_REF** element. The **Ref** attribute of that **OCCURRENCE_REF** element must contain the name of the **OCCURRENCE** definition that appropriately transforms the form.

Image data which is repeated in multiple page descriptions should be converted into a **REUSABLE_OBJECT**. The process for this is similar to what is needed for forms except that the image data should be stored in TIFF or JPEG instead of in PostScript.

The transformation matrix in each **OCCURRENCE** defines the changes to the Current Transform Matrix (CTM) made at the point of reference to the image. Therefore, if the original code was:

```
300 400 translate
50 50 scale
100 200 1 [100 0 0 -200 0 200] decodeimage image
<image data>
```

The TIFF image would therefore contain 100 by 200 pixels at an x and y resolution of 72 dots per inch. Assuming the TIFF image data was stored in image1.tiff the **REUSABLE_OBJECT** definition would be:

```
  <REUSABLE_OBJECT>
  <OBJECT Position=”0 0”>
     <SOURCE Format=”image/tiff” Dimensions=”100 200”>
        <EXTERNAL_DATA Src=”image1.tif”/>
     </SOURCE>
     <VIEW>
        <TRANSFORM Matrix=”0.01 0 0 0.005 0 0”/>
     </VIEW>
  </OBJECT>
  <OCCURRENCE_LIST>
     <OCCURRENCE Name=”image1”>
        <VIEW>
           <TRANSFORM Matrix=”50 0 0 -50 300 450”/>
        </VIEW>
     </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

The value for **OBJECT/VIEW/TRANSFORM/@Matrix** can be found by executing the following PostScript fragment:

```
[100 0 0 200 0 0] matrix invertmatrix ==
```

By applying this transformation, the image is scaled to always be 1 by 1 point in size.

The value for **OCCURRENCE/VIEW/TRANSFORM/@Matrix** should be equal to:

$$[100\ 0\ 0\ -200\ 0\ 200] \times [100\ 0\ 0\ 200\ 0\ 0]^{-1} \times [50\ 0\ 0\ 50\ 300\ 400]$$

in which the matrix [50 0 0 50 300 400] describes the changes to the CTM. This transformation combines the effect of the image-matrix for the `image` operator and the transformations to the CTM into a single transformation mapping the 1x1 pt image to its correct place on the page.

The same transformation matrix can be found by executing the following PostScript fragment:

```
[100 0 0 -200 0 200]
[100 0 0 200 0 0]
matrix invertmatrix matrix concatmatrix
matrix currentmatrix matrix invertmatrix
300 400 translate
50 50 scale
matrix currentmatrix exch matrix concatmatrix matrix concatmatrix ==
```

For images that only differ in location on the page, the resulting transformation matrices will only differ in the last two numbers of those matrices. In such cases, the same **OCCURRENCE** definition can be referenced.

It is recommended to define the **OCCURRENCE** with a transformation matrix in which the last two numbers are set to 0. The *Position* attribute of the **MARK** element that is used to position the image (using **OCCURRENCE_REF**) can then be set to the last two numbers of the transformation matrix calculated for that image.

PDF to PPML application notes

Today many PDF based workflows exist. PPML/GA allows PDF content to be repurposed and optimized for printing. In PDF files that are created using VDP software, elements repeated on multiple pages are often stored as a Form XObject in PDF.

To effectively use PDF as content for a PPML dataset, the PDF output needs to be modified as follows:

- each Form or Image XObject must be output as a PDF page,
- each PDF page that references a Form or Image XObject should be split into multiple pages. Each page containing a part of the original page before or after a call to draw a Form or Image XObject.

PDF allows translucent form and Image objects to be defined. PPML does not support translucency therefore such Form and Image objects cannot be optimized in PPML and must remain a Form or Image XObject in the PDF content data.

For each Form or Image XObject, a **REUSABLE_OBJECT** is created. Using an **EXTERNAL_DATA_ARRAY** element the newly generated PDF page containing the Form or Image content is referenced.

PPML only allows a **REUSABLE_OBJECT** to be positioned (not rotated or scaled) and an **OCCURRENCE** must be defined for each transformation applied to the Form or Image in the original PDF content. References to Form and Image XObjects are replaced by **OCCURRENCE_REF** elements that reference the **OCCURRENCE** that applies the appropriate transformation. Each part of the original PDF pages is placed onto a PPML page using a **MARK** with an **EXTERNAL_DATA_ARRAY** element to reference that PDF page.

A PPML/GA compliant dataset can then be constructed as follows (using A4 paper):

```
<PPML>
   <CONFORMANCE SubSet=”GA” Level=”1”/>
   <PAGE_DESIGN TrimBox=”0 0 595 842”/>
   <JOB>
   <REUSABLE_OBJECT>
      <OBJECT Position=”0 0”>
         <SOURCE Format=”application/pdf”
            Dimensions=”[form-1-urx] [form-1-ury]”
            ClippingBox=”[form-1-bounding-box]”>
            <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”1”
IndexUsage=”Mulitple”/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCE_LIST>
         <OCCURRENCE Name=”form-1-view-1”>
```

```
                <VIEW>
                    <TRANSFORM Matrix=”[form-1-transform-1]”/>
                </VIEW>
            </OCCURRENCE>
                :
            <OCCURRENCE Name=”form-1-view-2”>
                <VIEW>
                    <TRANSFORM Matrix=”[form-1-transform-2]”/>
                </VIEW>
            </OCCURRENCE>
        </OCCURRENCE_LIST>
    </REUSABLE_OBJECT>
        :
    <REUSABLE_OBJECT>
        <OBJECT Position=”0 0”>
            <SOURCE Format=”application/pdf”
                Dimensions=”[form-k-urx] [form-k-ury]”
                ClippingBox=”[form-k-bounding-box]”>
                <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”k”
IndexUsage=”Multiple”/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCE_LIST>
            <OCCURRENCE Name=”form-k-view-1”>
                <VIEW>
                    <TRANSFORM Matrix=”[form-k-transform-1]”/>
                </VIEW>
            </OCCURRENCE>
                :
            <OCCURRENCE Name=”form-k-view-2”>
                <VIEW>
                    <TRANSFORM Matrix=”[form-k-transform-2]”/>
                </VIEW>
            </OCCURRENCE>
        </OCCURRENCE_LIST>
    </REUSABLE_OBJECT>
    <DOCUMENT>
            <PAGE>
                <MARK Position=”0 0”>
                    <OBJECT Position=”0 0”>
```

```
                  <SOURCE Format=”application/pdf”
                     Dimensions=”595 842”>
                     ClippingBox=”[bbox-page-1-1]”>
                     <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”k+1”
IndexUsage=”Multiple”/>
                  </SOURCE>
               </OBJECT>
            </MARK>
            <MARK Position=”[x] [y]”>
               <OCCURRENCE_REF Ref=”form-1-view-1”/>
            </MARK>
            <MARK Position=”0 0”>
               <OBJECT Position=”0 0”>
                  <SOURCE Format=”application/pdf”
                     Dimensions=”595 842”
                     ClippingBox=”[bbox-page-1-2]”>
                     <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”k+2”
IndexUsage=”Multiple”/>
                  </SOURCE>
               </OBJECT>
            </MARK>
         </PAGE>
            :
         <PAGE>
            <MARK Position=”0 0”>
               <OBJECT Position=”0 0”>
                  <SOURCE Format=”application/pdf”
                      Dimensions=”595 842”>
                     <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”k+n”
IndexUsage=”Multiple/>
                  </SOURCE>
               </OBJECT>
            </MARK>
         </PAGE>
      </DOCUMENT>
   </JOB>
</PPML>
```

PPML with PDF-based content can also be converted into PPML/VDX compliant datasets. More PDF-based examples can be found in the PPML/VDX application notes[1].

[1] Committee for Graphic Arts Technologies Standards*. Application Notes for CGATS.20 (PPML/VDX), August 204.*

Page picking

PPML may also be used to re-use existing PDF assets. Using an **EXTERNAL_DATA_ARRAY** individual PDF pages may be taken from multiple PDFs and combined into a new print file. Page numbers, headers and footers may be added using dynamically generated PostScript or PDF content. Simple applications include reversing the output to simulate face-down/face-up output.

The following PPML dataset shows how to concatenate pages from two PDF files into a single document.

```
<PPML>
   <CONFORMANCE SubSet=”GA” Level=”1”/>
   <PAGE_DESIGN TrimBox=”0 0 595 842”/>
   <JOB>
   <DOCUMENT>
          <PAGE>
             <MARK Position=”0 0”>
                 <OBJECT Position=”0 0”>
                    <SOURCE Format=”application/pdf”
                       Dimensions=”[bbox-page-1-1]”>
                       <EXTERNAL_DATA_ARRAY Src=”file-1.pdf” Index=”1”
IndexUsage=”Multiple”/>
                    </SOURCE>
                 </OBJECT>
             </MARK>
          </PAGE>
          <PAGE>
             <MARK Position=”0 0”>
                 <OBJECT Position=”0 0”>
                    <SOURCE Format=”application/pdf”
                       Dimensions=”[bbox-page-1-2]”>
                       <EXTERNAL_DATA_ARRAY Src=”file-1.pdf” Index=”2”
IndexUsage=”Multiple”/>
                    </SOURCE>
                 </OBJECT>
             </MARK>
          </PAGE>
          <PAGE>
             <MARK Position=”0 0”>
                 <OBJECT Position=”0 0”>
                    <SOURCE Format=”application/pdf”
```

```
                        Dimensions=”[bbox-page-2-1]”>
                        <EXTERNAL_DATA_ARRAY Src=”file-2.pdf” Index=”2”
IndexUsage=”Multiple”/>
                    </SOURCE>
                 </OBJECT>
             </MARK>
          </PAGE>
       </DOCUMENT>
   </JOB>
</PPML>
```

Impositioning

Multiple PDF pages may be combined into a PPML sheet allowing complex imposition schemes to be implemented using PPML datasets without the need to modify the original content data.

The following PPML dataset shows how to implement 2-up A4 onto an A3 sheet:

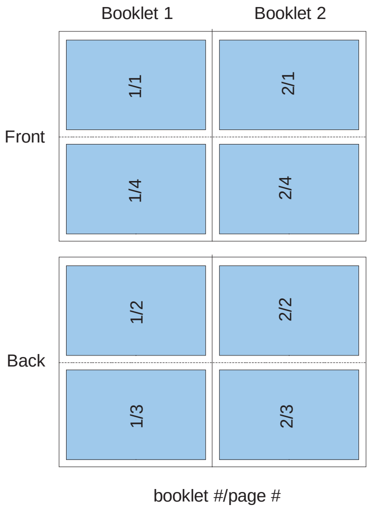
```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”1”/>
    <PAGE_DESIGN TrimBox=”0 0 1190 842”/>
    <JOB>
    <DOCUMENT>
            <PAGE>
                <MARK Position=”0 0”>
                    <OBJECT Position=”0 0”>
                        <SOURCE Format=”application/pdf”
                            Dimensions=”[bbox-page-1]”>
                            <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”1”
IndexUsage=”Multiple”/>
                        </SOURCE>
                    </OBJECT>
                </MARK>
                <MARK Position=”595 0”>
                    <OBJECT Position=”0 0”>
                        <SOURCE Format=”application/pdf”
                            Dimensions=”[bbox-page-2]”>
                            <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”2”
IndexUsage=”Multiple”/>
                        </SOURCE>
                    </OBJECT>
                </MARK>
            </PAGE>
                :
            <PAGE>
                <MARK Position=”0 0”>
                    <OBJECT Position=”0 0”>
                        <SOURCE Format=”application/pdf”
                            Dimensions=”[bbox-page-n-1]”>
                            <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”n-1”
IndexUsage=”Multiple”/>
```

```
                        </SOURCE>
                    </OBJECT>
                </MARK>
                <MARK Position=”595 0”>
                    <OBJECT Position=”0 0”>
                        <SOURCE Format=”application/pdf”
                            Dimensions=”[bbox-page-n]”>
                            <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”n”
IndexUsage=”Multiple”/>
                        </SOURCE>
                    </OBJECT>
                </MARK>
            </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

The following PPML dataset shows how to implement 4-up A5 onto an A3 sheet, where each of the A5 pages is rotated for folding into A5 booklets:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”1”/>
    <PAGE_DESIGN TrimBox=”0 0 1190 842”/>
    <JOB>
    <DOCUMENT>
            <PAGE>
                <MARK Position=”0 0”>
                    <OBJECT Position=”0 0”>
                        <SOURCE Format=”application/pdf”
                            Dimensions=”[bbox-page-1]”>
                            <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”1”
IndexUsage=”Multiple”/>
                        </SOURCE>
                    </OBJECT>
                    <VIEW>
                        <TRANSFORM Matrix=”0 1 1 0 0 0”/>
                    </VIEW>
                </MARK>
                <MARK Position=”0 421”>
                    <OBJECT Position=”0 0”>
                        <SOURCE Format=”application/pdf”
                            Dimensions=”[bbox-page-2]”>
```

```
                     <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”2”
IndexUsage=”Multiple”/>
                  </SOURCE>
               <VIEW>
                  <TRANSFORM Matrix=”0 1 -1 0 0 0”/>
               </VIEW>
               </OBJECT>
            </MARK>
            <MARK Position=”595 0”>
               <OBJECT Position=”0 0”>
                  <SOURCE Format=”application/pdf”
                     Dimensions=”[bbox-page-3]”>
                     <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”3”
IndexUsage=”Multiple”/>
                  </SOURCE>
               </OBJECT>
               <VIEW>
                  <TRANSFORM Matrix=”0 1 1 0 0 0”/>
               </VIEW>
            </MARK>
            <MARK Position=”595 421”>
               <OBJECT Position=”0 0”>
                  <SOURCE Format=”application/pdf”
                     Dimensions=”[bbox-page-2]”>
                     <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”4”
IndexUsage=”Multiple”/>
                  </SOURCE>
               <VIEW>
                  <TRANSFORM Matrix=”0 1 -1 0 0 0”/>
               </VIEW>
               </OBJECT>
            </MARK>
         </PAGE>
            :
      </DOCUMENT>
   </JOB>
</PPML>
```



Example 1 Imposed pages

Annex A
(informative)
Revision history

Version 1.0, January 2011

Initial release.

Index

Exhibit 2
to Gauthier Declaration

PERSONALIZED PRINT MARKUP LANGUAGE

PPML Templates: Methods and Workflows

Creating print-ready PPML document streams automatically and efficiently, directly from raw data

Functional Specification

PPML Templating Specification

Version 1.0 – December 12, 2002
The PPML Working Group

PODi
the Digital Printing initiative

PPML

The Personalized Print Markup Language

http://www.podi.org

Feedback and Developer Participation

PODi welcomes feedback on this specification, and offers the following services to support widespread adoption of the specification:

- ### Specification Updates

  The PPML specification, and related specifications, are distributed free of charge. If you are a developer who will be implementing the PPML standard, you should subscribe to the free PPML updates and tech note service.

  Additional PPML features are already planned, and some aspects of the specification are likely to be refined as development proceeds. The spec document itself will be updated, and technical notes will be published containing clarifications, implementation notes, and so on.

- ### Developer Support web site

  If you are a software or hardware developer interested in supporting PPML, you can register to participate in the PPML Developers discussion group. At present, there is no charge for this service.

To participate in the PPML initiative in any of the above ways, send an email to ppmlinfo@podi.org.

PODi

The Digital Printing Initiative

Web: http://www.podi.org

Table of Contents

Chapter 1: Introduction

1.1 Purpose

The purpose of this specification is to enable PPML workflows that have higher value, by automating the generation of certain document streams directly from raw data, and to encourage growth of high-value workflows by defining an industry standard way of doing it.

The motivation for PPML templating is to enable very long runs of the most valuable type of digital printing – personalized content – without the substantial overhead traditionally required to generate and transmit large amounts of repetitive data. As shown by the examples in the Appendix, when properly applied PPML templating can reduce the amount of data required for such a print run by two or three orders of magnitude.

Not all PPML applications are suited to templating. PPML is capable of a very wide range of digital print applications; templating is appropriate for those where parts of the PPML stream can be replaced with variable content as described in this document.

1.2 Prerequisite reading

This document presumes that the reader is familiar with the basic concepts of PPML, the Personalized Print Markup Language. Readers who don't know PPML can use this document to learn the basic idea of templating. But to create actual templated projects, it's necessary to fully understand PPML. The PPML specification is freely available at http://www.PODi.org.

1.3 PPML as part of a larger workflow

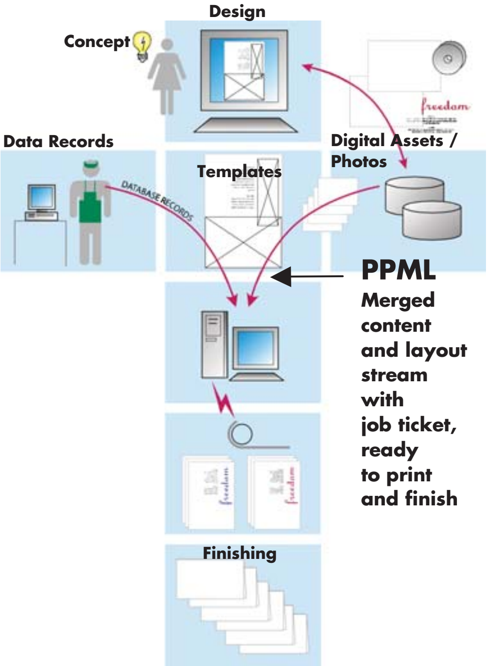
1.3.1 Introduction

PPML is the Personalized Print Markup Language, an XML-based metalanguage for digital print applications. Developed by the members of PODi, the Digital Print initiative, PPML was introduced to the market in February 2000 and has received statements of support from almost all vendors of digital print equipment and related application software. For more information see http://www.podi.org.

A major strength of PPML is its XML syntax. This means the entire range of XML data processing tools can be used to generate, analyze, and process PPML data. PPML templating, as described in this document, takes advantage of this.

1.3.2 The PPML Architecture

PPML provides an open, XML-based architecture for digital print projects. It was first introduced to the market at the worldwide "drupa" exhibition in Dusseldorf in May, 2000, and has become the first widely-adopted print stream based entirely on an open standard.



PPML can be generated by any workflow, automated or operator-controlled. Its natural affinity for data-driven applications means that the workflow concept shown here is common for PPML applications:

- The project concept is converted to a page design template by an operator at a workstation. This may be done using graphical tools or by creating logical expressions in a templating language.
- To create a print run, data records and digital assets (such as photos) are blended using the template. The result is a stream of fully marked-up PPML documents.
- The PPML is fed to a digital print system, which processes the pages, prints the documents, and (in suitably equipped systems) feeds them to automated finishing equipment.

The PPML specification is format-neutral, allowing content data to be supplied in any format that a machine supports. As such, it is not limited to the graphic arts or any other application segment, and its design can be extended in response to new opportunities and applications that are recognized by member companies and PODi management.

Since version 1.0, PPML has been extended beyond being a content stream. Today PPML provides a complete workflow architecture:

- **Device-independent document content.** Documents can be encoded into PPML without knowledge of the specific device that will print them.
- **Open to all content formats.** PPML does not specify content format; it provides metadata about document structure and layout. Thus, it is immediately adaptable to any new application that may arise that uses a different content format from those previously associated with digital print. Among other things, this means any new PPML-based print system can easily be driven by all PPML-producing software, even if the new system is in a market that's not normally associated with digital print.
- **Device-independent job ticketing.** As defined in this document, common processing parameters such as media selection, RIPping parameters, and finishing instructions can be inserted into the PPML stream without knowledge of the specific device that will print them. The open PPML ticketing architecture allows this to be done using the JDF standard or other ticket formats.
- **A design for packaging content and job ticket for reliable transport.** An appendix to the PPML specification defines VDP rules for creating a PPML print project (job

content, layout, and job ticket) on one system and transporting it to another, where it can be unpacked and printed reliably, even in cross-platform applications.

Because the entire PPML architecture is XML-based, all of the content, structure, and job ticket data in a PPML project can be generated, manipulated, extracted, subsetted, and processed in any way that is supported by common XML data tools. In addition, metadata and other types of non-printing content can be embedded in PPML through the use of the XML namespace mechanisms. This sort of flexibility and versatility has never before been available in a print stream, illustrating the power of the PPML design.

Further extension of the PPML architecture is being planned. For more information, contact PODi at info@podi.org.

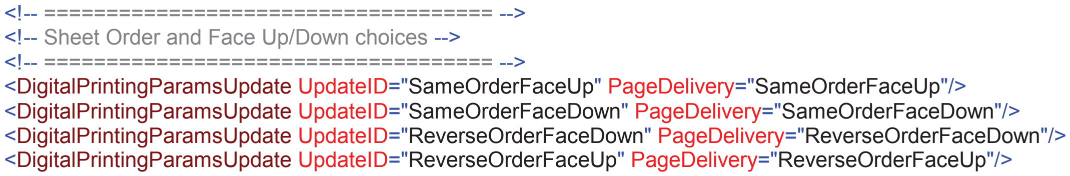
1.4 Scope of this specification

This specification describes an extension to PPML for use with scripting technologies such as XSLT. It does not fully describe XSLT, nor the PPML language itself. Rather, it only describes a method of accomplishing the transform into PPML.

1.5 Notation used in this document

The following typographic notation is used in this document.

- PPML code excerpts, element names, and attributes: `Letter Gothic`
- XML code samples are shown as displayed and formatted in the XMLSpy Integrated Development Environment, for instance:

  ```
  <!-- ==================================== -->
  <!-- Sheet Order and Face Up/Down choices -->
  <!-- ==================================== -->
  <DigitalPrintingParamsUpdate UpdateID="SameOrderFaceUp" PageDelivery="SameOrderFaceUp"/>
  <DigitalPrintingParamsUpdate UpdateID="SameOrderFaceDown" PageDelivery="SameOrderFaceDown"/>
  <DigitalPrintingParamsUpdate UpdateID="ReverseOrderFaceDown" PageDelivery="ReverseOrderFaceDown"/>
  <DigitalPrintingParamsUpdate UpdateID="ReverseOrderFaceUp" PageDelivery="ReverseOrderFaceUp"/>
  ```

  (In the body of this specification, no special formatting is applied to JDF element names.)
- The vertical bar character signifies the logical OR operator: |
  For instance, "`SOURCE | OCCURRENCE_REF`" means "`SOURCE` or `OCCURRENCE_REF`".
- Because many PPML element names are common English words, it is often convenient and accurate to use them conversationally. In this document, when an element name appears in text *not* in Courier, but with Initial Capitals, it is specifically referring to the PPML item that bears that name. When it appears with no capitalization, the word is being used with no special PPML significance. Example:

  The `SOURCE` element contains one or more component files.
  In an `OBJECT` element, the Source may contain data in any of several formats.
  Customers may submit image data that was gathered from a number of different sources.

  - In tables of XML attributes, when the data type is Number or Integer, a multiplication sign indicates a string of numbers separated by spaces. For instance, "Number ×4" indicates that the value of the attribute should be four numbers, such as "`1.234 2.0 3 4.567`."

1.6 Definitions

1.6.1 General PPML-related terms

Chapter 3 of the PPML Specification, "Terminology and Basic Concepts," defines basic terms regarding PPML document structure and workflow, including:

- **PPML Producer** (or simply "Producer") is anything that generates PPML files. This may be a standalone application, a system-level driver, or anything else.
- **PPML Consumer** (or simply "Consumer") is typically a RIP or DFE (digital front end to a digital printing device), but it may be any other device (or process or system) that reads and interprets PPML files. See the PPML specification for details on Consumer functionality. In particular, note that not all Consumers are required to support all features.
  Note that a PPML Consumer may also be a PPML Producer. For instance, an application could read PPML files, interpret their contents, modify the content or structure, and produce new PPML files.
- **Project** is all activities involving both the initial setup phase and the subsequent production runs. A Project is an on-going activity, consisting of multiple Jobs, as opposed to a conventional print job which is typically produced once and archived.
- **Dataset:** a `PPML` element, typically containing one or more Jobs and/or Reusable Object definitions and related elements required to process them.
- **Job** is the collection of activities and data to fulfill a single personalized printing work order, or to prepare the templates, objects, etc. that will later be used in fulfilling production work orders. In personalized printing, a Job is part of a Project.
  NOTE: the PPML language includes a `<JOB>` element with specific meaning in the hierarchical structure of PPML. (In PPML 2.1 `DOCUMENT_SET` is preferred; its meaning is identical.) In this document, "Job" (capitalized) refers to a PPML `<JOB>` element; "job" (lowercase) is informal, with no special meaning relative to PPML. For instance, it's correct to say "The supervisor asked Pat to run job #482, which contained three PPML Jobs."

The following terms are also used in this document:

- **Imposition** is the process of positioning page images on sheets of paper in the printer (or in a digital printing press), as part of the process of producing finished documents. See Chapter 6 of the PPML Specification.
- **Print Originator:** the person (or group) for whom a project is being produced – the person who conceived it and/or decided what they want the finished result (Product) to look like. The print originator knows what the desired end product is, and may have no knowledge of process, i.e. how the job will be produced.
- **Production Shop:** traditionally this term refers to the people who produce a print job, including their equip ment, software, procedures, and the physical facility itself. This may be a department of a company, a separate business, or any other entity. In this document, the term

refers to any entity that performs the work of such a Production Shop. See also "Workflow" under JDF, below.

- **RIP:** Raster Image Processor – an image processing system that reads data expressed in a PDL such as PostScript® and converts it to a raster image.
- **Format Processor:** the component of a PPML Consumer that processes objects submitted in a particular input data format. Examples: a PostScript RIP, or a module that can directly process an image format such as JPEG or TIFF.
- **PPML Job Ticket:** the data required to produce a set of printed documents, beyond the document content and layout specified in PPML. The ticket may be external to the PPML dataset or may be embedded in it, within a PPML `TICKET` element.

1.6.2 Terms related to Templating

- **PPML Template Producer** (or simply "Template Producer") is anything that generates a PPML Template. This may be a human editing a file, a standalone application, or anything else.
- **PPML Template Consumer** (or simply "Template Consumer") is anything that reads and executes `PPMLT` elements as defined in this document. PPML Template Consumers are PPML Producers as defined above: they generate PPML code.

  It is expected, but not required, that many PPML Consumers will also include built-in Template Consumer capability. But the Template Consumer may also be a separate production step, physically distinct from the PPML Consumer. In that case, the Template Consumer would read the PPMLT elements and produce an ordinary stream of PPML documents, which the PPML Consumer would process the same as if they didn't come from templating.
- **Template:** In the context of PPML Templating, the template is a prototype PPML document which has been modified to enable varying the content using a scripting language such as XSLT.
- **Data records:** the input data that will be merged with the template to generate the stream of Instance Documents. Example: the recipient's name, address, and last product purchased.
- **Style sheet:** In the context of PPML Templating this refers to XSLT style sheets. An XSLT style sheet contains the transformation instructions that define how Data Records are merged with the Template. For further information, see the XSLT web site.

1.7 Requirements

The PPML Templating workflow was developed to meet the following requirements:

1.7.1 Must not interfere with existing PPML Consumers in non-template applications

1.7.2 Support multiple formats for the data list

- XML, both simple and complex

- Comma-separated
- Line data

1.7.3 Compatible with PPML Requirements

PPML Templating shall not restrict or interfere with any existing requirements for the PPML datastream itself. In particular, it shall follow the PPML requirements for streaming; page independence, locatability, segmentability; manufacturing information; and variable document length.

1.7.4 Flexible workflow: Allow template and data to be transmitted in a single package or separately, which enables reusability of the data and the template

- Allow sending just a template to the Template Consumer for later use
- Allow sending just the data, to be used with a previously sent template. Note that the same template can be used repeatedly with different sets of data. This, in fact, is the most productive way to use variable data, since it amortizes the project's initial setup cost over a much longer project lifespan.

Chapter 2: Applications of PPML Templating

2.1 Introduction

There is a range of possibilities for how the processing of a print job can be divided among system components to achieve the benefits of a templating workflow. This chapter presents a framework for understanding the choices, and the benefits and limitations of what templating can do, so that workflow designers can make well-informed choices and make optimal use of this technology for their chosen applications.

In general the amount that can be achieved through templating depends on the amount of information available to the PPML Template Producer.

For instance, typically a template processor will be very good at performing routine, iterative operations like inserting data into a pre-defined layout. But features like text composition and detection of reusable content may not be part of a Template Consumer’s abilities, in which case they would need to be handled elsewhere in the workflow, before data is transmitted to the Template Consumer. The PPML Template Producer may or may not have access to information generated during those processes, such as the height of copy blocks after they’re composed.

2.2 Benefits of templating

PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer).

In a fully optimized PPML Template workflow, virtually nothing remains to be downloaded at print time except the data itself. Very little data is generated, very little data is transmitted, and very little data is processed at the receiving end. As shown by the examples in the appendix, the result can be substantial savings.

In addition, by directly transforming variable data into a structured print stream (PPML), PPML Templating enables new workflows independent of constraints of traditional graphic arts system.

2.3 How templating differs from conventional workflows

2.3.1 Anatomy of a variable print project

To understand the impact of templating, it’s useful to understand the environment in which it’s designed to be used: the generation of streams of personalized documents.

Templating involves a simple but elegant shift in the location of one portion of the personalized print workflow: the point where the variable data is merged with the page layout.

As suggested above, personalized documents are typically characterized by having a common and repetitive structure, with content that varies from individual to individual. There are many different ways to generate such a stream. Examples range from simple office mail-merge (e.g. form letters) to highly sophisticated, automated, multi-variable page design systems which can completely vary the content and even the page layout. Regardless of the details, all these workflows must, at one time or another, accomplish two primary tasks:

- **Job setup**, in which the form letter or other document is designed
- **Production runs**, in which batches of documents are generated and printed.

More specifically, all personalized print workflows must handle the following tasks.

- **Job Setup**
    - **Design of the basic document** that will be personalized: decisions about content and layout.
    - **Decisions about how layout and content will vary**, depending on the variable data. These decisions are often referred to in VDP applications as rules. The rules are initially created by humans, and they must be encoded in some electronic form and stored somewhere, to enable automated production.
    - **Creation of the reusable content objects** and downloading to the print system.
- **Production runs**
    - **Selecting the variable data** for each print run. *This is unchanged in Templating.*
    - **Executing the VDP rules** for each recipient, to generating the personalized documents. *In Templating, this is done at a different time and place.*
    - **Generating the output code** to make the target print system produce the documents. *In PPML templating, this is merged with the previous step.*

The follow sections describe typical workflows, with and without templating. Many variations on these workflows exist in current practice in the industry; the purpose of this discussion is to illustrate the change that PPML Templating creates.

2.3.2 Conventional Variable Data workflow

- **Job setup:**
    - The page producer application ("Producer") generates:
        - A document template including
            - The basic document, including designation of what areas may change. This is typically stored in the Producer's native format.

        - The rules for creating instance documents, e.g. where to insert variable text and how to select reusable content elements. Like the layout, the rules are typically stored in the Producer's native format.

      - The definitions of PPML Reusable Objects

    - The Reusable Object definitions are typically transmitted to the Consumer for processing, proof-printing, approval, and storage.

- **Production runs:**
    - The database generates the variable data records for a print run. For instance, this may include the name, address, and product interest of each recipient.
    - The Producer opens the stored template and merges it with the variable data for each variable data record, using the template's VDP rules. The result is a stream of personalized documents.
    - The Producer uses its PPML driver to convert the personalized documents to a stream of PPML Instance Documents.
    - The PPML Instance Documents are transmitted from the Producer to the Consumer. (This may be done in many ways: by direct cable connection from Producer to Consumer, or by copying the PPML dataset to a CD, or compressing it into an archive such as a Zip file, etc.)
    - The Consumer executes the PPML, resulting in a stream of printed documents.

The next section illustrates the similarities – and differences – in a PPML Templating workflow.

2.3.3 PPML Template workflow

The initial document design is identical and the end result (the printed output) is identical. But several of the steps are moved to a different point in the workflow.

- **Job setup** is very similar to the conventional workflow, but the VDP rules are encoded in a scripting language (e.g. XSLT syntax) in a prototype PPML document. (See next chapter for more information on XSLT.) The template is downloaded to the PPML Template Consumer.

  Note: "PPML Template Consumer" refers to the part of the workflow that executes the template. This functionality may be resident inside the PPML Consumer, or it may take place in a pre-processing step. This has no bearing on the content of this specification.

- **Production:**
    - The database generates the same variable data records for a print run. The database records are packaged in the `DATA` element described in this specification.
    - The data is transmitted to the Template Consumer.

- *New:* The Template Consumer opens the transmitted PPML Template dataset, accesses the template (which was sent earlier), and executes the template's instructions once for each variable data record. The result is a stream of PPML Instance Documents. Note:
    - Functionally, this merging of template and data is the same as what the Producer did in the conventional workflow: the VDP rules are applied to each recipient's data. But in this workflow, the rules are executed inside the Template Consumer.
    - With this workflow it is not necessary for the Producer to generate an intermediate stream of personalized documents in its native format, and then convert each one to PPML for transmission to the PPML Consumer. Instead, the PPML is generated directly from the raw data. Also, if the PPML Template is processed inside the PPML Consumer, it avoids redundant generation and transmission of the basic document structure.
- At this point the stream of PPML code looks exactly the same as it did in the conventional workflow described above.
- Thus, the print stream that the PPML Consumer executes is identical in both workflows, and the resulting output is identical. But the result was produced with a bare minimum of data handling.

2.4 Examples

Templating can offer significant savings in processing on the originator side, and in data volume in the overall workflow, but there are some situations in which it may not be appropriate, or in which more sophisticated work may be necessary to create the transformations, involving more advanced applications of scripting technologies. This section presents some issues for consideration by designers of templating workflows.

The principle that "the amount templating can accomplish is a function of how much data is sent" is illustrated by the following discussion of basic and advanced applications.

2.4.1 Basic applications

In the simple case, PPML Templating is best suited to applications where the layout, page count, line breaks, and reusable objects are known in advance, so they don't need to be computed as part of executing the template. Examples include (but are by no means limited to):

- Simple direct mail pieces, like a "mail merge" application, including variable text or images.
- Point-of-purchase materials, using PPML reusable content with variable pricing etc.
- Photo album pages, with a static layout and a reusable page border but with references to different JPEG files on every page.

2.4.2 Advanced Applications

Advanced workflow developers, with strong knowledge of both PPML and a scripting technology, can create highly complex and sophisticated documents with this technology. For example:

- Personalized catalogs with variable sized items on different pages
- Customized financial statements that include data-driven graphics and individually selected offers or advisory content
- Collateral on demand, such as brochures assembled on the fly in response to a user's selections on a Web site.

2.4.3 Considerations in the design of templating workflows

As shown by the above examples, a broad range of workflows can be designed using PPML Templating, by choosing appropriate tools and components based on the type of output desired and the data available for input to the process. Factors to consider include:

- How will the workflow manipulate and transform the data? (For instance, can a particular scripting language accomplish the necessary conversions?)
- How will the workflow convert the raw uncomposed data into objects that can be placed on pages using PPML? (For instance, if text needs to be reflowed into composed paragraphs, how will that be achieved?)

For assistance in configuring workflows for a particular need, consult your vendor of PPML Templating systems.

Chapter 3: XML, Scripting, and XSLT

3.1 Introduction

This chapter presents an introduction to scripting, particularly XML and XSLT, as they apply to PPML templating. For detailed information, see the resources listed here.

3.2 Scripting technologies for PPML Templates

This specification defines an architecture that can be used with any sort of scripting technology. As described below, XSLT is a natural choice, because it is based on XML, as is PPML itself.

However, PPML Templating applications are not limited to XSLT. For instance, another language that is expected to be well-suited to templating is PERL.

3.3 XML

PPML is an application of XML, the Extensible Mark-up Language. Readers who are not familiar with XML are directed to these resources:

- XML.ORG (http://www.xml.org) is an industry web portal operated by OASIS, the Organization for the Advancement of Structured Information Standards.
- OASIS's "The SGML/XML Web Page" (http://www.oasis-open.org/cover/sgml-xml.html) contains many excellent links to reference information.
- "The XML.commune" (http://www.xml.com) is a collaborative partnership between Seybold Publications and Songline Studios, an affiliate of O'Reilly & Associates. The site includes Tim Bray's excellent annotated version of the XML syntax recommendation.
- Project Cool XML Zone (http://www.projectcool.com/developer/xmlz/) is one of the best sites for developers, with a fairly good introduction to the basics of XML.

3.4 XSLT

3.4.1 Overview of what XSLT does

XSLT http://www.w3.org/TR/xslt is one of the two parts of XSL, the Extensible Style Language http://www.w3.org/Style/XSL. It transforms one XML file into another one. For instance, it can be used to transform data conforming to one DTD into a form conforming to another.

The expression language of XSLT is XPath.

3.4.2 High-level description

An XSLT processor does these things:

- It reads a set of instructions encoded in a "style sheet" file [.xsl] and builds a tree representation in memory. The term "style sheet" is historical. Although XSLT can indeed be used to insert formatting instructions, it is actually a general purpose XML transformation language.
- The style sheet includes two general things:
  - XSLT instructions, which tell the XSLT processor what data to locate in the input file(s) and how to manipulate and interpolate that data into the output
  - Literal text and non-XSLT XML
- The processor reads the default XML source file and builds a tree representation in memory. As processing continues, other source XML files may be opened but there is always one (and only one) available at the outset.
- Instructions in the style sheet tree direct the processor to generate a result tree in memory consisting of nodes created by copying the literal text and non-XSLT XML plus the result of executing other XSLT instructions. These other XSLT instructions create what are known as "result tree fragments", or RTFs, which are subsequently copied to the end result tree.
- Optionally, the processor serializes the result tree into an output file. The format of the output file may or may not be XML depending upon the output method specified in the XSLT script. Typical output methods are "xml", "html" and "text". Unless otherwise specified, the output method will be "xml".

There are many XSL processors. A commonly used one is Xalan (http://xml.apache.org/xalan-j/index.html). Saxon (http://saxon.sourceforge.net/) is another very popular XSL processor. A good list of XSL software tools is at http://xml.coverpages.org/xslSoftware.html

3.5 Format of an XSL template file

An XSL template file, or script, is a well formed XML file consisting of:

- XSLT instructions
- Non-XSLT XML
- Plain text

The non-XSLT XML and plain text constitute what are known as "Literal Result Elements" because they are copied to the result tree unmodifield.

One of the common XSLT instructions is `xsl:template`. The `xsl:template` instruction can be invoked by name or, more commonly, by matching some construct in the input XML file or files. The match is described using the XPath (http://www.w3.org/TR/xpath) XML vocabulary.

Each `xsl:template` itself contains XSLT instructions, non-XSLT XML and plain text. This content is processed when the template matches a construct in the input XML.

Another common XSLT instruction is `xsl:apply-templates`. The `xsl:apply-templates` instruction selects a set of nodes from the input XML and looks for the `xsl:template` instruction that best matches each, if any. That `xsl:template` instruction is then executed with the matching source node as an implicit parameter.

Every source XML file has an implicit, anonymous node at its head. This node, known as the "root node," is the parent of the top-level node in the source XML document. This node would be matched by an XPath match description of "/".

3.6 Literal Result Element as Stylesheet

A special form of XSLT scripts is known as a "Literal Result Element as Stylesheet". In this special form, no `xsl:template` instructions are present. Rather, an arbitrary XML file can have the XSLT namespace defined on it and be passed to an XSLT processor as the template, or script, file.

The entire XML file is treated as if it were a single Literal Result Element located within an `xsl:template` that matched the root node, i.e., match="/". As such, any non-XSLT XML and its plain text content are copied to the result tree verbatim. Any XSLT instructions found are evaluated as encountered and the outcome is also placed in the result tree.

By associating the XSLT namespace with a PPML document, it is possible to add XSLT instructions and pass that document as a stylesheet to an XSLT processor.

3.7 General sequence of events in XSL

This describes the general sequence, which will be illustrated by the example below.

- The XSL processor program is started with three parameters:
    - The initial input XML data file (e.g. the customer records)
    - The template file to read (the XSL instructions for how to generate an output file)
    - The name of the output file to create

  Sample command line (using the Xalan processor `xsl.exe`):

  ```
      xsl  mydata.xml  ppml.xsl  ppmlout.xml
  ```

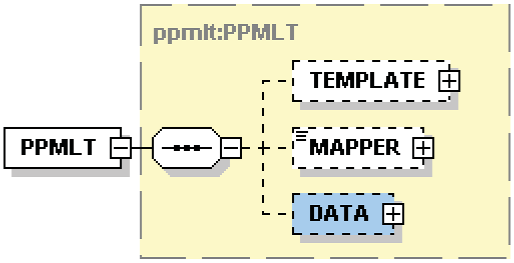
- The processor begins by reading the template file and building a tree representation of that file in memory. Note that the template file must be a well-formed XML file.
- The processor then reads the source XML file (the variable data records), which also must be well-formed, and builds a tree representation of that file in memory.
- The processor searches for the xsl:template that matches the XPath expression "/", i.e., the root node, and applies that template to the node. If no such template exists, the default template is one that applies templates to the children of the current node, recursively. As described above, an XSLT file employing a Literal Result Element as Stylesheet is considered to be the content of a template matching the root node.

- Any non-XSLT data, text or XML, is copied to the result tree.
- Upon encountering an `<xsl:...` instruction (these are highlighted in blue in the examples in this document), the instruction is evaluated and the result is placed on the result tree. These instructions may locate data in the source XML file, manipulate that data or both.

Chapter 4: Structure of a PPML Templating Project

4.1 Overview

A PPML Templating project requires two components for TEMPLATE and DATA. An optional third element, DATA_MAPPER, may connect them:



- The TEMPLATE element contains a special kind of PPML document that is ready to be merged with variable data as described in this specification.
- The DATA element contains the variable data records for one particular print run. The most common formats for the data are expected to be XML and comma-separated values.
- An optional DATA_MAPPER element can define data format conversions.

It is expected that these job components will often be delivered from Producer to Consumer in a single package (see the PPML specification's "packaging" appendix) but that process is not required. The template and data may be created and delivered at any time, independent of each other.

4.2 Element content: Internal vs. External Data

As described in the PPML Specification, PPML document content and resources may be contained directly in the XML element that requires them, or they may be external, identified by a URI reference. For PPML Templating, this method is used for the TEMPLATE element and the DATA element. Below are simple examples of a TEMPLATE element constructed both ways.

4.2.1 External Data example (reference to an external file)

```
<TEMPLATE ...>
   <EXTERNAL_DATA  Src=”Template472.xml” />
</TEMPLATE>
```

External data may or may not be included in the same PPML Template package. See Appendix D of the PPML Specification for discussion of the advantages of delivering a complete package, and for recommended restrictions on the value of the Src attribute to ensure cross-platform portability.

In applications where the PPML Template Producer wants to ensure that a specific version of the referenced data is used, the Producer can provide a checksum.

4.2.2 Internal Data example

The Internal Data method inserts the content of the template into the TEMPLATE element, verbatim.

```
<TEMPLATE ...>
   <INTERNAL_DATA>
      (The template data goes here – the content of the file “Template472.xml”
       that was referenced by EXTERNAL_DATA in the example above)
   </INTERNAL_DATA>
</TEMPLATE>
```

4.2.3 Example of referencing content downloaded earlier

Content may also be downloaded and saved with a name for reference later. Example:

```
<TEMPLATE Name=”472” Environment=”TestFiles”>
   <EXTERNAL_DATA  Src=”Template472.xml” />
</TEMPLATE>
```

The content can later be referenced as follows:

```
<TEMPLATE_REF Name=”472” Environment=”TestFiles”/>
```

4.3 The <PPMLT> Element

4.3.1 Description

The PPMLT element is the top level, encompassing all components of the PPML templating job or this portion of it. (For instance, the template may be transmitted in one PPMLT element, and the data may be transmitted in a separate PPMLT element.)

4.3.2 Model

```
PPMLT       (
               (DATA
                  |  TEMPLATE, ((DATA_MAPPER | DATA_MAPPER_REF)?,
                        (DATA | DATA_REF))?
                  |  TEMPLATE_REF, ((DATA_MAPPER | DATA_MAPPER_REF)?,
                        (DATA | DATA_REF))
                  |  DATA_MAPPER
               )
            )
```

4.3.3 Attributes

None.

4.3.4 Result of processing a PPMLT element

A PPMLT element can contain, directly or by reference, a template or data or both. The expected behavior of the Template Consumer for each case is:

4.3.4.1 Template only

Extract the template data from the PPMLT (the INTERNAL_DATA content or the referenced EXTERNAL_DATA) and save it locally in the Template Consumer.

4.3.4.2 Data only

Extract the data from the PPMLT DATA element (specifically, from the INTERNAL_DATA content or the referenced EXTERNAL_DATA inside the DATA element) and save it locally in the Template Consumer.

4.3.4.3 Both template and data are identified

Execute the specified template, operating on the specified input data.

4.4 The <TEMPLATE> Element

4.4.1 Description

The TEMPLATE element identifies the prototype PPML document which will be used to generate the PPML Instance Documents.

The TEMPLATE element can contain either an INTERNAL_DATA or an EXTERNAL_DATA element. This means new template instructions (layout and VDP rules) can be downloaded within the PPML Template instance (INTERNAL_DATA) or the template can be downloaded in advance and referenced with EXTERNAL_DATA.

Allowing the template to be either internal or external to the dataset provides flexibility that can be useful in a variety of situations. See examles below.

An optional DATA_STRUCTURE element can be included, to describe the structure of the variable data expected by the template. This description can be used to allow validation by the receiving system or by any intermediate processing tools. (Note that the data contained in the DATA element may not be XML; if not, it must be converted to XML before merging with the template. See discussion below.)

4.4.2 Model

```
TEMPLATE     (DATA_STRUCTURE?, (INTERNAL_DATA | EXTERNAL_DATA))
```

4.4.3 Attributes

Attribute	Required /Optional	Type	Description
Format	Required	String	The format of this template. Must be a valid MIME type.
Name	Optional	String	Name to be used when referring to this template. The name must be unique within the template's environment. If this attribute is used, the Template Consumer must save this template, making it available for reference by subsequent PPMLT elements via a TEMPLATE_REF element.
Environment	Optional	String	Specifies the environment in which the template's name exists. (There is no default environment.) Required if the Name attribute is used.

4.4.4 Context

The TEMPLATE element occurs only within a PPMLT element.

4.4.5 Application note: saving templates for later use

When a PPML Template Consumer processes a TEMPLATE element with a Name attribute (and therefore also with an Environment attribute), the Template Consumer is required to save the template for later use, along with the value of the Format attribute.

Management of templates installed in a Template Consumer is not specified in this document. The developer of a Template Consumer is responsible for providing a way to manage them, e.g. deleting templates that are no longer needed.

4.4.6 Examples

Example 1: `TEMPLATE` element contains an External Data reference. This method would typically be used in case of repetitive projects, in which the template stays resident at the receiving system; in such cases, the PPML Template dataset transmits the variable data (in the `DATA` element) and references the previously downloaded template file.

```
<PPMLT>
   <TEMPLATE Format="text/xslt+xml">
     <EXTERNAL_DATA  Src=”Project412.xsl”/>
   </TEMPLATE>
   <DATA>
     <INTERNAL_DATA>
        Variable data records go here
     </INTERNAL_DATA>
   </DATA>
</PPMLT>
```

Example 2: The template is contained in an `INTERNAL_DATA` element; the `DATA` element references data that was downloaded earlier. This approach could be used to generate a different document stream from a set of variable data records that were already sent earlier.

```
<PPMLT>
   <TEMPLATE>
     <INTERNAL_DATA>
        Template data goes here
     </INTERNAL_DATA>
   </TEMPLATE>
   <DATA>
     <EXTERNAL_DATA  Src=”Project412_2003-10-09.xml”/>
   </DATA>
</PPMLT>
```

Example 3: `TEMPLATE` and `DATA` elements both contain External Data references. The following is a complete PPML Templating file, sufficient to cause the printing of a batch of personalized documents by associating a previously downloaded template with a previously downloaded variable data file.

```
<PPMLT>
   <TEMPLATE>
     <EXTERNAL_DATA  Src=”Project412.xsl”/>
   </TEMPLATE>
   <DATA>
     <EXTERNAL_DATA  Src=”Project412_2003-10-09.xml”/>
   </DATA>
</PPMLT>
```

4.5 The <TEMPLATE_REF> Element

4.5.1 Description

The `TEMPLATE_REF` element identifies, by reference, a template that has already been installed in the Template Consumer using a `TEMPLATE` element with the `Name` and `Environment` attributes.

4.5.2 Model

```
TEMPLATE_REF Empty
```

4.5.3 Attributes

Attribute	Required /Optional	Type	Description
Ref	Required	String	Name of the previously installed template. The name must be unique within the template's environment.
Environment	Required	String	Specifies the environment of the template's name. (There is no default environment.)
Checksum	Optional	String	Hexadecimal-encoded string, provided as a hint to the Template Consumer as an aid in identifying the template that was installed earlier. Template Consumers are not required to support this attribute.
ChecksumType	Optional	String	Identifies the type of checksum. If this attribute is present, the Checksum attribute must also be present. Default="MD5".

4.5.4 Context

The `TEMPLATE_REF` element occurs only within a `PPMLT` element.

4.6 The <DATA> Element

4.6.1 Description

The DATA element contains the database records to be merged with the template, to generate personalized Instance Documents. For a deeper discussion of handling different types of data, see Chapter 5: Data.

4.6.2 Model

```
DATA          (DATA_STRUCTURE?, (INTERNAL_DATA | EXTERNAL_DATA))
```

4.6.3 Attributes

Attribute	Required /Optional	Type	Description
Format	Required	String	The format of this data. Any valid MIME type.
Name	Optional	String	Name to be used when referring to this data . The name must be unique within the template's environment.
Environment	Optional	String	Specifies the environment in which the template's name should be available. (There is no default environment.) Required if the Name attribute is used.

4.6.4 Context

The DATA element occurs only within a PPMLT element.

4.6.5 Application note: saving data files for later use

When a PPML Template Consumer processes a DATA element with a Name attribute (and therefore also with an Environment attribute), the Template Consumer is required to save the data for later use, along with the value of the Format attribute.

Management of data files installed in a Template Consumer is not specified in this document. The developer of a Template Consumer is responsible for providing a way to manage them, e.g. deleting files that are no longer needed.

4.6.6 Character set conversion

The PPML Template Consumer may need to convert the data from the character set specified in the enclosed data element to the character set expected by the Template Consumer's template language. The only character set explicitly supported by XML parsers is Unicode; Template Consumers are not required to support any other character set.

Example: XSLT expects the Unicode character set. If the data in the DATA element is encoded in EBCDIC, the Template Consumer must map the incoming EBCDIC characters to the corresponding Unicode characters.

4.7 The <DATA_REF> Element

4.7.1 Description

The DATA_REF element identifies, by reference, a data file that has already been installed in the Template Consumer using a DATA element with the Name and Environment attributes.

4.7.2 Model

```
DATA_REF      Empty
```

4.7.3 Attributes

Attribute	Required /Optional	Type	Description
Ref	Required	String	Name of the previously installed data file. The name must be unique within the environment.
Environment	Required	String	Specifies the environment of the name. (There is no default environment.)
Checksum	Optional	String	Hexadecimal-encoded string, provided as a hint to the Template Consumer as an aid in identifying the data file that was installed earlier. Template Consumers are not required to support this attribute.
ChecksumType	Optional	String	Identifies the type of checksum. If this attribute is present, the Checksum attribute must also be present. Default="MD5".

4.7.4 Context

The DATA_REF element occurs only within a PPMLT element.

4.8 The <EXTERNAL_DATA> Element

4.8.1 Description

An EXTERNAL_DATA element identifies, by location and access method, a single content datum (e.g. a template or data file).

The EXTERNAL_DATA type is inherited from the PPML specification, with the addition of the CharacterSet attribute. Any changes to this element in the PPML XML Schema will automatically propagate to the PPMLT schema.

4.8.2 Model

```
    EXTERNAL_DATA    EMPTY
```

4.8.3 Attributes

Attribute	Required /Optional	Type	Description
Src	Required	URI	URI (Uniform Resource Identifier) string identifying the external data. See RFC2396 for full details of URIs.[1] See also application note below.
Checksum	Optional	String	Hexadecimal-encoded string, provided as a hint to the Template Consumer. Template Consumers are not required to support this attribute.
ChecksumType	Optional	String	Identifies the type of checksum. If this attribute is present, the Checksum attribute must also be present. Default="MD5".
CharacterSet	Optional	String	Identifies the character set used in the referenced data. Default="UTF-8". See description under INTERNAL_DATA.

4.8.4 Context

EXTERNAL_DATA may occur within TEMPLATE and DATA.

4.8.5 Application note regarding URI

A PPML Template Consumer is not required to support any particular access protocol (for instance, HTTP), so a data emitter cannot be certain that URIs in EXTERNAL_DATA will be readable by an unknown Template Consumer. Therefore, if a data emitter wants to ensure that the template will be readable by any Template Consumer, INTERNAL_DATA should be used.

[1] RFC2396 is at http://www.ietf.org/rfc/rfc2396.txt. A good overview of URIs and URLs is at http://www.w3.org/Addressing/Overview.html.

4.9 The <INTERNAL_DATA> Element

4.9.1 Description

An `INTERNAL_DATA` element is the same as an `EXTERNAL_DATA` element except that it contains the actual data, instead of referring to it. Therefore it has no `Src` attribute.

4.9.2 Model

```
INTERNAL_DATA    ANY
```

4.9.3 Attributes

Attribute	Required /Optional	Type	Description
Encoding	Optional	Keyword	Encoding scheme of the data: `None` (default) or any encoding name registered with the Internet Assigned Numbers Authority (IANA).[2] However, note that Template Consumers are only required to support `Base64`.
CharacterSet	Optional	String	Specifies the character set of the decoded data. For use with text content or any other media type containing characters. Value: any character set name registered with the Internet Assigned Numbers Authority (IANA).[3] Default: the character set of the enclosing PPMLT file.
Label	Optional	String	Any arbitrary string to identify this element, for instance in case an error message is necessary.
Creator	Optional	String	Identifies the application that created this content.

4.9.4 Context

`INTERNAL_DATA` may occur within `TEMPLATE` and `DATA`.

[2] The valid encoding name strings are listed at http://www.isi.edu/in-notes/iana/assignments/transfer-encodings.

[3] The valid character set name strings are at http://www.isi.edu/in-notes/iana/assignments/character-sets.

4.10 The <DATA_STRUCTURE> Element

4.10.1 Description

The optional DATA_STRUCTURE element describes the structure of the data expected by the PPML Template script.

Note that any method of description is allowed as content of the DATA_STRUCTURE element, e.g. DTD, RELAX, XML Schema or even plain text.

In XML applications, the DATA_STRUCTURE inside TEMPLATE describes the format of the XML expected by that template. This XML may be provided directly or generated at the PPML Template Consumer from non-XML data.

The DATA_STRUCTURE inside DATA describes the structure of that DATA.

4.10.2 Model

```
DATA_STRUCTURE   (INTERNAL_DATA | EXTERNAL_DATA)
```

4.10.3 Attributes

Attribute	Required /Optional	Type	Description
Format	Required	String	The format of this description. Must be a valid MIME type.

4.10.4 Context

The DATA_STRUCTURE element occurs within a TEMPLATE or DATA element.

4.10.5 Notes

It is assumed that most, if not all, initial implementations of PPML Templating will use the DATA_STRUCTURE elements for documentation purposes only. However the information stored there can be used to automate the process of mating databases with templates. See also the DATA_MAPPER element.

4.11 The <DATA_MAPPER> element

4.11.1 Description

The optional DATA_MAPPER element contains a script designed to reformat the input data (specified in the DATA element) to the form expected by a PPML Template script. The result of applying DATA_MAPPER to the DATA becomes the input to TEMPLATE.

If DATA is sent separately from TEMPLATE, each may have an associated DATA_MAPPER element. This allows flexibility in configuring the workflow, for instance to adapt to the needs of individual customers or systems. Examples:

- The incoming data can be transformed into the format expected by the template
- A PPML Template Consumer system could have several DATA_MAPPER scripts available, to accommodate a variety of different incoming data formats.
- Both might be true. For instance, a DATA_MAPPER script may be included in the PPMLT element that includes the DATA element, which transforms the data into the format expected by another DATA_MAPPER script in the PPMLT element that contains the template. In this case, the DATA_MAPPER associated with DATA is applied first.

Optionally, the DATA_MAPPER element may also contain descriptions of the structure of the input and output data describing the input format expected by the script and the output that it will generate. Initial PPML Templating implementations may only use these elements for documentation – to document what format is expected as input to the XSLT script and what format the will output. However future implementations may consider making some intelligent use of the information conveyed in these descriptions.

4.11.2 Model

```
DATA_MAPPER   ((INPUT_DATA_STRUCTURE, OUTPUT_DATA_STRUCTURE)?,
              (INTERNAL_DATA | EXTERNAL_DATA))
```

4.11.3 Attributes

Attribute	Required /Optional	Type	Description
Format	Optional	String	The format of this template. Must be a valid MIME type.
Name	Optional	String	Name to be used when referring to this data mapper . The name must be unique within the environment.
Environment	Optional	String	Specifies the environment in which the mapper's name should be available. (There is no default environment.) Required if the Name attribute is used.

4.11.4 Context

DATA_MAPPER occurs only at the top level, in a PPMLT element.

4.11.5 Notes

If INPUT_DATA_STRUCTURE is present, then OUTPUT_DATA_STRUCTURE must follow it.

4.11.6 Application note: saving data mappers for later use

When a PPML Template Consumer processes a DATA_MAPPER element with a Name attribute (and therefore also with an Environment attribute), the Template Consumer is required to save the mapper for later use, along with the value of the Format attribute.

Management of data mappers installed in a Template Consumer is not specified in this document. The developer of a Template Consumer is responsible for providing a way to manage them, e.g. deleting files that are no longer needed.

4.12 The <DATA_MAPPER_REF> Element

4.12.1 Description

The DATA_MAPPER_REF element identifies, by reference, a data mapper that has already been installed in the Template Consumer using a DATA_MAPPER element with the Name and Environment attributes.

4.12.2 Model

```
DATA_MAPPER_REF  Empty
```

4.12.3 Attributes

Attribute	Required /Optional	Type	Description
Ref	Required	String	Name of the previously installed mapper. The name must be unique within the environment.
Environment	Required	String	Specifies the environment of the name. (There is no default environment.)
Checksum	Optional	String	Hexadecimal-encoded string, provided as a hint to the Template Consumer as an aid in identifying the mapper that was installed earlier. Template Consumers are not required to support this attribute.
ChecksumType	Optional	String	Identifies the type of checksum. If this attribute is present, the Checksum attribute must also be present. Default="MD5".

4.12.4 Context

The DATA_REF element occurs only within a PPMLT element.

4.13 The <INPUT_DATA_STRUCTURE> Element

4.13.1 Description

The `INPUT_DATA_STRUCTURE` element describes the data format of the data being converted by a `DATA_MAPPER` element. As noted in the previous section, this is provided on an "information only" basis – there is no requirement that the receiving system do anything with this information.

In the case of an XML database, the content of `INPUT_DATA_STRUCTURE` would be an XML Schema or DTD describing the format of that database.

4.13.2 Model

```
    INPUT_DATA_STRUCTURE    (INTERNAL_DATA | EXTERNAL_DATA)
```

4.13.3 Attributes

Attribute	Required /Optional	Type	Description
Format	Required	String	The format of this description.

4.13.4 Context

The `INPUT_DATA_STRUCTURE` element occurs only within a `DATA_MAPPER` element.

4.14 The <OUTPUT_DATA_STRUCTURE> Element

4.14.1 Description

Like INPUT_DATA_STRUCTURE, this is an optional "information only" element within DATA_MAPPER. It describes the data format of the output generated by the DATA_MAPPER's script.

4.14.2 Model

```
OUTPUT_DATA_STRUCTURE  (INTERNAL_DATA | EXTERNAL_DATA)
```

4.14.3 Attributes

Attribute	Required /Optional	Type	Description
Format	Required	String	The format of this description.

4.14.4 Context

The OUTPUT_DATA_STRUCTURE element occurs only within a DATA_MAPPER element.

Chapter 5: Data

5.1 Introduction

The most natural data format for input to PPML Templating is XML. However, some data sources (spreadsheets, legacy systems, etc) are not set up to export in XML format; instead, they typically output delimiter-separated values or plain unformatted line data.

This chapter describes an optional but standardized method of encoding record-oriented data into XML, in a PPML Template <R> and <F> structure, which is contained in a root <RECORDS> element. This information may be of use for developers of systems who wish to incorporate this ability into their PPML Template Consumer. A pre-processor could also be developed to convert the data to R/F format, placing the result into a separate file which can be referenced using EXTERNAL_DATA.

5.1.1 Delimiter-separated values (“DSV”)

In this data format, each individual record is on a separate line, and the fields are separated with a delimiter. Typically a comma is used, resulting in the “comma-separated values” format (CSV) that is commonly output by applications such as Microsoft Excel. A common alternative is tab-delimited data.

Converting DSV data to R/F format is trivial. Each line of input data (i.e. each record) becomes an <R> element, and each field value is placed into an <F> element. Example:

CSV data:

```
John,Watson,12 Main St.,Anywhere,NY,10021
Mary,Smith,47 Broadview,OurTown,NH,03079
```

Converted to R/F format:

```
<RECORDS>
    <R>
        <F>John</F>
        <F>Watson</F>
        <F>12 Main St.</F>
        <F>Anywhere</F>
        <F>NY</F>
        <F>10021</F>
    </R>
    <R>
        <F>Mary</F>
        <F>Smith</F>
        <F>47 Broadview</F>
        <F>OurTown</F>
        <F>NH</F>
        <F>03079</F>
    </R>
</RECORDS>
```

5.1.2 Line data

Some computer systems output data with no delimiters. Instead, fields are identified by their fixed column position.

This example shows the same data as in the previous section:

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
John        Watson    12 Main St.       Anywhere      NY     10021
Mary        Smith     47 Broadview      OurTown       NH     03079
```

In this example the text in columns 1-12 is the first field, columns 13-22 is the second field, etc. When converted to R/F format the result would be the same as the example above.

Many application tools are available to parse line data into fields.

5.1.3 Parameterizing these conversions

PPML Template workflow designers may have a choice regarding how to handle data formats that are essentially identical except for certain parameters. For instance tab-delimited data is essentially the same as comma-delimited.

Some workflows may find it more convenient to have separate Mapper scripts (section 4.11) for comma-delimited and tab-delimited cases; these scripts can easily be selected by a reference in EXTERNAL_DATA. Others may prefer to design a single script that handles all delimiter-separated files, using a parameter to identify what the delimiter is. In this case the script's parameters could be passed from the Producer to the script via arguments in the URI.

Example: a user might create a universal script for handling delimiter-separated values. The script might accept a parameter named “delim”. To use that script for files delimited with tabs (0x09), a URI to reference that script might be (the parameter is shaded for easy identification):

```
<DATA_MAPPER>
  <EXTERNAL_DATA  Src="MyDelimiterScript.xsl?delim=&#x09;">
</DATA_MAPPER>
```

When the same script is used to process a file delimited with commas (0x2C), the URI might be:

```
<DATA_MAPPER>
  <EXTERNAL_DATA  Src="MyDelimiterScript.xsl?delim=&#x2C;">
</DATA_MAPPER>
```

5.2 The <RECORDS> element

5.2.1 Description

The RECORDS element is the top level, providing the root node that contains all the R and F records.

5.2.2 Model

```
RECORDS       (R*)
```

5.2.3 Attributes

None.

5.2.4 Context

Within a PPMLT element, RECORDS occurs only in INTERNAL_DATA. RECORDS may also occur in a separate data file referenced by EXTERNAL_DATA.

5.3 The <R> element

5.3.1 Description

The R element contains one record of variable data, consisting of one or more F elements.

5.3.2 Model

```
   R                (F+)
```

5.3.3 Attributes

None.

5.3.4 Context

R occurs only in DATA.

5.4 The <F> element

5.4.1 Description

The F element contains one field of data within a record.

5.4.2 Models

```
    F               (#PCDATA)
```

5.4.3 Attributes

Attribute	Required /Optional	Type	Description
Name	Optional	String	The name of this field. This attribute is provided as a convenience for human readability, including cross-referencing to the original (non-XML) file. It may also be of use in workflows that require named fields.

5.4.4 Context

The F element occurs only inside R.

5.5 Very long data streams

When presented with very long data streams, such as hundreds of thousands of records, a tree-oriented scripting system such as XSLT can become resource-intensive (causing problems with speed or memory). The Xpath expressions within XSLT have random access to the entire XML data tree, so XSLT processing effectively requires reading the entire XML tree, and most XSLT processors have significant performance problems when the tree is large.

Two alternatives are available to avoid this problem:

- A data emitter can transmit the data in multiple, smaller PPMLT files. See example 1 below.
- PPML Template Consumers are allowed to break up the incoming data into smaller "chunks" at the boundary between record boundaries. For XML data, the Template Consumer is allowed to terminate the tree between immediate children of the root element. See example 2.
  Note: In applications where the template requires multiple records of input data per Instance Document, care must be taken to "chunk" between appropriate record groups. A DATA_MAPPER script can be used to pre-process the data for this purpose, as shown in example 3 below.

The following PPMLT file is used in the examples below.

```
<PPMLT>
 <TEMPLATE_REF  Env=”Myco”  Ref=”MyTemplate”/>
 <DATA>
   <INTERNAL_DATA>
     <RECORDS>
        <R><F>Record1.....</R>
        <R><F>Record2.....</R>
        <R><F>Record3.....</R>
        ....
        <R><F>Record99.....</R>
     </RECORDS>
   </INTERNAL_DATA>
 <DATA>
</PPMLT>
```

Example 1: Data emitter breaks the data into smaller PPMLT files

The data emitter closes off one PPMLT element and opens another one, so that the PPML Template Consumer processes a long job as several small ones. The lines highlighted in blue are inserted:

```
<PPMLT>
 <TEMPLATE_REF  Env=”Myco”  Ref=”MyTemplate”/>
 <DATA>
   <INTERNAL_DATA>
     <RECORDS>
        <R><F>Record1.....</R>
        <R><F>Record2.....</R>
        ....
        <R><F>Record50.....</R>
     </RECORDS>
   </INTERNAL_DATA>
 <DATA>
</PPMLT>
<PPMLT>
```

```
  <TEMPLATE_REF  Env="Myco"  Ref="MyTemplate"/>
  <DATA>
    <INTERNAL_DATA>
      <RECORDS>
         <R><F>Record51.....</R>
         ....
         <R><F>Record99.....</R>
      </RECORDS>
    </INTERNAL_DATA>
  <DATA>
</PPMLT>
```

Example 2: Template Consumer "chunks" the data

In this method the Template Consumer breaks the data appropriate boundaries. In XML data, this is at any immediate child of the root element. For the "R/F" record/field structure described in this specification, the root element is RECORDS and the immediate children are the R records, so the Template Consumer may terminate the tree after any R element and start a new tree with the next R element, as if another root element had been inserted.

With chunking, the template processor is invoked more than once, and sees different sets of data on the different invocations, for instance:

Invocation 1:

```
<RECORDS>
   <R><F>Record1.....</R>
   <R><F>Record2.....</R>
   <R><F>Record3.....</R>
   ....
   <R><F>Record50.....</R>
</RECORDS>
```

Invocation 2:

```
<RECORDS>
   <R><F>Record51.....</R>
   ....
   <R><F>Record99.....</R>
</RECORDS>
```

Example 3: Using a DATA_MAPPER to perform multi-record grouping

Some templates may require that several records of data be kept together (e.g. applications where the data for one customer occupies three records). In this case, chunking between any immediate child (an R element) could result in invalid grouping of records. This problem can be circumvented by using a Mapper script to group elements together by adding a parent XML element. For instance:

Original ungrouped data:

```
<RECORDS>
   <R><F>Record1.....</R>
   <R><F>Record2.....</R>
   <R><F>Record3.....</R>
   <R><F>Record4.....</R>
   <R><F>Record5.....</R>
   <R><F>Record6.....</R>
   ....
   <R><F>Record97.....</R>
   <R><F>Record98.....</R>
   <R><F>Record99.....</R>
</RECORDS>
```

After processing by an appropriate Mapper:

In this example, the Template Producer provided a Mapper script that inserts a parent `CUSTOMER` element around sets of three records. ("Customer" is an arbitrary name in this example.) When that Mapper script is executed by the Template Consumer, the above data is transformed into the following structure. The immediate child of `RECORDS` is now `CUSTOMER`, so chunking will break at safe boundaries:

```
<RECORDS>
   <CUSTOMER>
      <R><F>Record1.....</R>
      <R><F>Record2.....</R>
      <R><F>Record3.....</R>
   </CUSTOMER>
   <CUSTOMER>
      <R><F>Record4.....</R>
      <R><F>Record5.....</R>
      <R><F>Record6.....</R>
   </CUSTOMER>
      ...
   <CUSTOMER>
      <R><F>Record97.....</R>
      <R><F>Record98.....</R>
      <R><F>Record99.....</R>
   <CUSTOMER>
</RECORDS>
```

Appendix A: Sample Application

A.1 Introduction

This chapter contains a complete PPML Templating job, with both the TEMPLATE and the DATA expressed as INTERNAL_DATA. It then shows how the job can be made increasingly more efficient by storing more and more repetitive content in the Template Consumer: first the PPML Reusable Object definitions and then the document template.

A.2 Example 1: PPML Templating code, including Reusable Object definitions, complete PPML Template and Data Mapper, and data records

This code totals 14.5k; the 25 records of customer data at the end add 3.5k, for a total of 18k.

```
<?xml version="1.0" encoding="UTF-8"?>
<PPMLT xmlns="http://www.podi.org/ppmlt/ppmlt001.xsd">
   <TEMPLATE Format="application/xslt+xml">
      <INTERNAL_DATA>
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
           xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:svg="svg" version="1.0">
         <xsl:output indent="yes"/>
         <xsl:strip-space elements="*"/>
         <xsl:template match="/">
            <!-- Copyright Atlas Software BV -->
```

The color-highlighted block below contains the prototype PPML file. The yellow portion will be output once; it contains the start-of-job information, including definitions of reusable content, sheet layout, etc. The blue-shaded portion contains an XSLT "for-each", so it will be output repeatedly, once for each customer record, as explained below.

```
            <PPML>
               <DOCUMENT_SET Label="Job Number 1">
                 <IMPOSITION Name="Imporef">
                    <SIGNATURE Nrows="1" Ncols="1">
                       <CELL Row="1" Col="1" Face="Up" PageOrder="s"/>
                    </SIGNATURE>
                 </IMPOSITION>
                 <PRINT_LAYOUT>
                    <PAGE_LAYOUT TrimBox="0 0 612 792"/>
                    <SHEET_LAYOUT Hsize="612" Vsize="792">
                     <IMPOSITION_REF Name="Imporef"/>
                    </SHEET_LAYOUT>
                 </PRINT_LAYOUT>
                 <PRIVATE_INFO Creator="Xeikon" Identifier="MasterVDF"/>
                 <REUSABLE_OBJECT>
                     <OBJECT Position="0 0">
                       <SOURCE Format="application/postscript" Dimensions="612 792">
                         <EXTERNAL_DATA Src="OldsMobile.eps"/>
                       </SOURCE>
                     </OBJECT>
                     <OCCURRENCE_LIST>
```

```
       <OCCURRENCE Name="XMASTER_ OldsMobile.eps_1 0 0 1 0 0"
          Environment="Demo">
         <VIEW>
            <TRANSFORM Matrix="1 0 0 1 0 0"/>
         </VIEW>
       </OCCURRENCE>
    </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="612 792">
        <INTERNAL_DATA>
           gsave
           /psmPaintRect { gsave newpath 4 2 roll moveto 1 index 0
              rlineto 0
           exch rlineto neg 0 rlineto fill grestore } bind def
           0 0 0 0 setcmykcolor
           0 0 612 792 psmPaintRect
           grestore
        </INTERNAL_DATA>
      </SOURCE>
   </OBJECT>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="XMASTER_BackForeground_1_1" Environment="Demo"/>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="165.4 15">
        <INTERNAL_DATA>
           gsave
           /psmPaintRect { gsave newpath 4 2 roll moveto 1 index 0
              rlineto 0
           exch rlineto neg 0 rlineto fill grestore } bind def
           0 0 0 1 setcmykcolor
           0 0 165.421707153 15.0005950928 psmPaintRect
           grestore
        </INTERNAL_DATA>
      </SOURCE>
   </OBJECT>
   <OCCURRENCE_LIST>
     <OCCURRENCE Name="BackForeground_2_1" Environment="Demo"/>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="165.4 15">
        <INTERNAL_DATA>
           gsave
           /psmPaintRect { gsave newpath 4 2 roll moveto 1 index 0
              rlineto 0
           exch rlineto neg 0 rlineto fill grestore } bind def
           0 0 0 1 setcmykcolor
           0 0 165.421707153 15.0004425049 psmPaintRect
           grestore
        </INTERNAL_DATA>
      </SOURCE>
   </OBJECT>
   <OCCURRENCE_LIST>
     <OCCURRENCE Name="BackForeground_3_1" Environment="Demo"/>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="165.4 14">
        <INTERNAL_DATA>
           gsave
           /psmPaintRect { gsave newpath 4 2 roll moveto 1 index 0
              rlineto 0
           exch rlineto neg 0 rlineto fill grestore } bind def
```

```
            0 0 0 1 setcmykcolor
            0 0 165.421707153 14.0002288818 psmPaintRect
            grestore
         </INTERNAL_DATA>
      </SOURCE>
   </OBJECT>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="BackForeground_4_1" Environment="Demo"/>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="164.4 15">
         <INTERNAL_DATA>
            gsave
            /psmPaintRect { gsave newpath 4 2 roll moveto 1 index 0
               rlineto 0
            exch rlineto neg 0 rlineto fill grestore } bind def
            0 0 0 1 setcmykcolor
            0 0 164.422149658 15.0004425049 psmPaintRect
            grestore
         </INTERNAL_DATA>
      </SOURCE>
   </OBJECT>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="BackForeground_5_1" Environment="Demo"/>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="190.5 12.5">
         <INTERNAL_DATA>
            gsave
            /psmPaintRect { gsave newpath 4 2 roll moveto 1 index 0
               rlineto 0
            exch rlineto neg 0 rlineto fill grestore } bind def
            0 0 0 1 setcmykcolor
            0 0 190.499694824 12.5 psmPaintRect
            grestore
         </INTERNAL_DATA>
      </SOURCE>
   </OBJECT>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="BackForeground_6_1" Environment="Demo"/>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="191 94">
         <EXTERNAL_DATA Src="PURPLE"/>
      </SOURCE>
   </OBJECT>
   <VIEW>
      <CLIP_RECT Rectangle="0.04066 0.227 191 93.77"/>
   </VIEW>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="PURPLE_1 0 0 1 -0.04066 -0.227"
         Environment="Demo">
         <VIEW>
            <TRANSFORM Matrix="1 0 0 1 -0.04066 -0.227"/>
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
     <SOURCE Format="application/postscript" Dimensions="191 94">
        <EXTERNAL_DATA Src="BLUE"/>
     </SOURCE>
   </OBJECT>
   <VIEW>
      <CLIP_RECT Rectangle="0.04066 0.227 191 93.77"/>
   </VIEW>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="BLUE_1 0 0 1 -0.04066 -0.227" Environment="Demo">
         <VIEW>
            <TRANSFORM Matrix="1 0 0 1 -0.04066 -0.227"/>
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
     <SOURCE Format="application/postscript" Dimensions="191 94">
        <EXTERNAL_DATA Src="SILVER"/>
     </SOURCE>
   </OBJECT>
   <VIEW>
     <CLIP_RECT Rectangle="0.04066 0.227 191 93.77"/>
   </VIEW>
   <OCCURRENCE_LIST>
     <OCCURRENCE Name="SILVER_1 0 0 1 -0.04066 -0.227" Environment="Demo">
     <VIEW>
        <TRANSFORM Matrix="1 0 0 1 -0.04066 -0.227"/>
     </VIEW>
     </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
     <SOURCE Format="application/postscript" Dimensions="191 94">
        <EXTERNAL_DATA Src="GREENGRAY"/>
     </SOURCE>
   </OBJECT>
   <VIEW>
     <CLIP_RECT Rectangle="0.04066 0.227 191 93.77"/>
   </VIEW>
   <OCCURRENCE_LIST>
     <OCCURRENCE Name="GREEN/GRAY_1 0 0 1 -0.04066 -0.227"
         Environment="Demo">
        <VIEW>
           <TRANSFORM Matrix="1 0 0 1 -0.04066 -0.227"/>
        </VIEW>
     </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="191 94">
         <EXTERNAL_DATA Src="BLACK"/>
      </SOURCE>
   </OBJECT>
   <VIEW>
      <CLIP_RECT Rectangle="0.04066 0.227 191 93.77"/>
   </VIEW>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="BLACK_1 0 0 1 -0.04066 -0.227" Environment="Demo">
      <VIEW>
         <TRANSFORM Matrix="1 0 0 1 -0.04066 -0.227"/>
      </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="191 94">
         <EXTERNAL_DATA Src="GOLD"/>
      </SOURCE>
   </OBJECT>
   <VIEW>
```

```
      <CLIP_RECT Rectangle="0.04066 0.227 191 93.77"/>
   </VIEW>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="GOLD_1 0 0 1 -0.04066 -0.227" Environment="Demo">
         <VIEW>
            <TRANSFORM Matrix="1 0 0 1 -0.04066 -0.227"/>
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
     <SOURCE Format="application/postscript" Dimensions="191 94">
        <EXTERNAL_DATA Src="RED"/>
     </SOURCE>
   </OBJECT>
   <VIEW>
     <CLIP_RECT Rectangle="0.04066 0.227 191 93.77"/>
   </VIEW>
   <OCCURRENCE_LIST>
     <OCCURRENCE Name="RED_1 0 0 1 -0.04066 -0.227" Environment="Demo">
        <VIEW>
           <TRANSFORM Matrix="1 0 0 1 -0.04066 -0.227"/>
        </VIEW>
     </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position="0 0">
      <SOURCE Format="application/postscript" Dimensions="190.9 93.55">
         <INTERNAL_DATA>
            gsave
            /psmPaintRect { gsave newpath 4 2 roll moveto 1 index 0
               rlineto 0
            exch rlineto neg 0 rlineto fill grestore } bind def
            0 0 0 0 setcmykcolor
            0 0 190.918685913 93.5459136963 psmPaintRect
            grestore
         </INTERNAL_DATA>
      </SOURCE>
   </OBJECT>
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="BackForeground_7_1" Environment="Demo"/>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

The following blue-shaded copy will be output once for each XML CUSTOMER *element. In PPML without templating, the blue portion (approximately 4600 bytes) would be output once for each customer record.*

```
<xsl:for-each select="//CUSTOMER">
    <DOCUMENT>
      <PAGE>
        <MARK Position="0 0">
           <OCCURRENCE_REF Ref="XMASTER_BackForeground_1_1"
              Environment="Demo"/>
        </MARK>
        <MARK Position="0 0">
           <OCCURRENCE_REF Ref="XMASTER_ OldsMobile.eps_1 0 0 1 0 0"
              Environment="Demo"/>
        </MARK>
        <MARK Position="334 605">
           <OCCURRENCE_REF Ref="BackForeground_2_1" Environment="Demo"/>
        </MARK>
        <MARK Position="334 605">
           <OBJECT Position="0 0">
              <SOURCE Format="image/svg-xml" Dimensions="165 15">
                <INTERNAL_DATA>
                <svg:svg width="165pt" height="15pt">
```

```
          <svg:text x="82.5pt" y="10pt" font-family="Helvetica" font-
             size="10pt" word-spacing="1.294pt" letter-spacing=".129pt"
             text-anchor="middle" fill="rgb(255,255,255)">
             <xsl:value-of select="NAME"/>
          </svg:text>
        </svg:svg>
        </INTERNAL_DATA>
      </SOURCE>
   <VIEW>
      <TRANSFORM Matrix="1 0 0 1 0 0"/>
   </VIEW>
</OBJECT>
</MARK>
<MARK Position="334 548.5">
   <OCCURRENCE_REF Ref="BackForeground_3_1" Environment="Demo"/>
</MARK>
<MARK Position="334 548.5">
   <OBJECT Position="0 0">
      <SOURCE Format="image/svg-xml " Dimensions="165 15">
        <INTERNAL_DATA>
          <svg:svg width="165pt" height="15pt">
             <svg:text x="82.5pt" y="10pt" font-family="Helvetica"
                font-size="10pt" word-spacing="1.021pt" letter-
                spacing="0.102pt" text-anchor="middle"
                fill="rgb(255,255,255)">
                <xsl:value-of select="STREET"/>
             </svg:text>
          </svg:svg>
        </INTERNAL_DATA>
      </SOURCE>
      <VIEW>
        <TRANSFORM Matrix="1 0 0 1 0 0"/>
      </VIEW>
   </OBJECT>
</MARK>
<MARK Position="334 494">
   <OCCURRENCE_REF Ref="BackForeground_4_1" Environment="Demo"/>
</MARK>
<MARK Position="334 494">
   <OBJECT Position="0 0">
      <SOURCE Format="image/svg-xml " Dimensions="165 14">
         <INTERNAL_DATA>
           <svg:svg width="165pt" height="14pt">
              <svg:text x="82.5pt" y="9.5pt" font-family="Helvetica"
                 font-size="10pt" letter-spacing=".341pt" text-
                 anchor="middle" fill="rgb(255,255,255)">
                 <xsl:value-of select="PHONE"/>
              </svg:text>
           </svg:svg>
         </INTERNAL_DATA>
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="1 0 0 1 0 0"/>
      </VIEW>
   </OBJECT>
</MARK>
<MARK Position="334 438">
   <OCCURRENCE_REF Ref="BackForeground_5_1" Environment="Demo"/>
</MARK>
<MARK Position="334 438">
   <OBJECT Position="0 0">
      <SOURCE Format="image/svg-xml " Dimensions="164 15">
         <INTERNAL_DATA>
           <svg:svg width="164pt" height="15pt">
              <svg:text x="82pt" y="9.75pt" font-family="Helvetica"
                 font-size="9pt" letter-spacing="0.05pt" text-
                 anchor="middle" fill="rgb(255,255,255)">
                 <xsl:value-of select="EMAIL"/>
              </svg:text>
           </svg:svg>
```

```
                </INTERNAL_DATA>
              </SOURCE>
              <VIEW>
                <TRANSFORM Matrix="1 0 0 1 0 0"/>
              </VIEW>
           </OBJECT>
        </MARK>
        <MARK Position="84 582.5">
           <OCCURRENCE_REF Ref="BackForeground_6_1" Environment="Demo"/>
        </MARK>
        <MARK Position="84 582.5">
           <OBJECT Position="0 0">
              <SOURCE Format="image/svg-xml " Dimensions="190 13">
               <INTERNAL_DATA>
                  <svg:svg width="190pt" height="13pt">
                     <svg:text x="95pt" y="8.5pt" font-family="Helvetica" font-
                        size="8pt" word-spacing="1.789pt" letter-
                        spacing="0.179pt" text-anchor="middle"
                        fill="rgb(255,255,255)">
                        <xsl:value-of select="DESCRIPTION"/>
                     </svg:text>
                  </svg:svg>
               </INTERNAL_DATA>
              </SOURCE>
              <VIEW>
                <TRANSFORM Matrix="1 0 0 1 0 0"/>
              </VIEW>
           </OBJECT>
        </MARK>
        <MARK Position="84 598.2">
           <OCCURRENCE_REF Ref="BackForeground_7_1" Environment="Demo"/>
        </MARK>
        <MARK Position="84 598.2">
           <OCCURRENCE_REF Environment="Demo">
              <xsl:attribute name="Ref">
              <xsl:value-of select="IMAGE"/>
              <xsl:text>_1 0 0 1 -0.04066 -0.227</xsl:text>
              </xsl:attribute>
           </OCCURRENCE_REF>
        </MARK>
      </PAGE>
    </DOCUMENT>
</xsl:for-each>
```

Note for comparison: In PPML without templating, each record would add another copy of the DOCUMENT *element shown in blue above. This would lengthen the file by about 4.5k per record. In this example with 25 records, the PPML code would be 122k longer, for a total of approximately 140k. If it were fully shown in this specification, this example without templating would be 56 pages long.*

```
           </DOCUMENT_SET>
        </PPML>
      </xsl:template>
     </xsl:stylesheet>
   </INTERNAL_DATA>
</TEMPLATE>
```

The data mapper element is the same in the first two versions of this example.

```
<DATA_MAPPER Format="application/xslt+xml">
   <INTERNAL_DATA>
      <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
         xmlns:fo="http://www.w3.org/1999/XSL/Format">
        <xsl:output indent="yes"/>
        <xsl:template match="R">
           <CUSTOMER>
              <NAME>
                <xsl:value-of select="F[1]"/>
              </NAME>
```

```
             <STREET>
               <xsl:value-of select="F[2]"/>
             </STREET>
             <PHONE>
               <xsl:value-of select="F[3]"/>
             </PHONE>
             <EMAIL>
               <xsl:value-of select="F[4]"/>
             </EMAIL>
             <DESCRIPTION>
               <xsl:value-of select="F[5]"/>
             </DESCRIPTION>
             <IMAGE>
               <xsl:value-of select="F[6]"/>
             </IMAGE>
          </CUSTOMER>
        </xsl:template>
        <xsl:template match="/">
          <CUSTOMERS>
             <xsl:apply-templates/>
          </CUSTOMERS>
       </xsl:template>
     </xsl:stylesheet>
   </INTERNAL_DATA>
</DATA_MAPPER>
```

The Data element is the same in all versions of this example.

```
<DATA Format="application/xml">
  <INTERNAL_DATA>
      <RECORDS>
        <R><F>Cynthia Proctor</F><F>625 Missouri Street</F><F>510-372-7500</F>
           <F>dcgraphicdesigns@hotmail.com</F><F>1998 Purple Intrigue</F>
           <F>PURPLE</F></R>
        <R><F>Dr. Loose</F><F>Whoville</F><F>123-345-5678</F>
           <F>Loose@whoville.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
        <R><F>Henry Polard</F><F>33 World Trade Blvd.</F><F>650-855-9367</F>
           <F>polard@wenet.net</F><F>1998 Silver Intrigue</F><F>SILVER</F></R>
        <R><F>Al Joshua</F><F>4567 My Way</F><F>123-456-789</F>
           <F>ajoshua@psmail.com</F><F>1998 Silver Intrigue</F><F>SILVER</F></R>
        <R><F>Michelle Walker</F><F>860 36th Ave.</F><F>415/831-1019</F>
           <F>shelwalker@aol.com</F><F>1998 Green/gray Intrigue</F>
           <F>GREEN/GRAY</F></R>
        <R><F>Craig Kohler</F><F>860 36th Ave.</F><F>415/831/1019</F>
           <F>craig.kohler@schwab.com</F><F>1998 Black Intrigue</F><F>BLACK</F></R>
        <R><F>Ken Griffith</F><F>34286 Quartz St.</F><F>510-796-4975</F>
           <F>ken_griffith@splashtech.com</F><F>1998 White Intrigue</F>
           <F>WHITE</F></R>
        <R><F>Harry Raaphorst</F><F>Buys Ballotstraat 17-19</F><F>31-341-426700</F>
           <F>harry.raaphorst@atlasssoftware.nl</F><F>1998 Blue
           Intrigue</F><F>BLUE</F></R>
        <R><F>Michael Barnes</F><F>The Maxwell Company</F><F>415-123-4567</F>
           <F>maxwell@edu</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
        <R><F>Gregg Fox</F><F>200 Canal View Blvd. 831</F><F>716-427-4262</F>
           <F>gregg_fox@mc.xerox.com</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
        <R><F>Paul Lorton, Jr</F><F>1265 Altschul Av.</F><F>650-854-
           2406</F><F>lorton@usfca.edu</F><F>1998 Red Intrigue</F><F>RED</F></R>
        <R><F>Linda Jackson</F><F>405 - 1263 Barclay Street</F><F>604-844-2253</F>
           <F>linda_jackson@splashtech.com</F><F>1998 Black Intrigue</F> <F>BLACK</F>
           </R>
        <R><F>Denis Severson</F><F>3400 Hillview Ave.</F><F>650-813-7158</F>
           <F>severson@parc.xerox.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
        <R><F>Jindong Chen</F><F>3400 Hillview Ave, PAHV 12</F><F>650-813-7338</F>
           <F>jchen@parc.xerox.com</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
        <R><F>Gary Roth</F><F>8758 Wescott Court</F><F>619-484-3226</F>
           <F>gary_roth@splashtech.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
        <R><F>Susan Prischmann</F><F>3930 North Pinegrove, Apt.</F><F>312-849-4361</F>
           <F>sprischmann@currentassets.com</F><F>1998 Green/charcoal
           Intrigue</F><F>GREENCHARCOAL</F></R>
```

```
            <R><F>Sue Hoffmann</F><F>2000 Powell Street</F><F>657-1777</F>
               <F>sue_hoffmann@thenet.com</F> <F>1998 Red Intrigue</F><F>RED</F></R>
            <R><F>Rick Placak</F><F>130 So. Center</F><F>702-329-
               3145</F><F>rplacak@thenet.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
            <R><F>Betsy Pryser</F><F>1130 N. Dearborn, #1603</F><F>312-397-
               9250</F><F>epryser@ix.netcom.com</F><F>1998 Silver
               Intrigue</F><F>SILVER</F></R>
            <R><F>Mike Mayo</F><F>124 West Oxmoor</F><F>205-942-
               2222</F><F>jmmayo@worldnet.att.net</F><F>1998 Gold
               Intrigue</F><F>GOLD</F></R>
            <R><F>Armand Petri</F><F>1508 Blackhawk Drive</F><F>408 735
               9482</F><F>apetri@aol.com</F><F>1998 Purple Intrigue</F><F>PURPLE</F></R>
            <R><F>Ted DiSilvestre</F><F>333 W. San Carlos St.</F><F>408-536-
               6508</F><F>tdisilve@adobe.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
            <R><F>Dean Griswold</F><F>6947 West Oak Ct.</F><F>916-725-7739</F>
               <F>griswold@ix.netcom.com</F><F>1998 Green/gray Intrigue</F>
               <F>GREEN/GRAY</F></R>
            <R><F>John Doe</F><F>46 Nowhere Street</F><F>654-321-0987</F>
               <F>john_doe@nowhere.com</F><F>1998 Black Intrigue</F><F>BLACK</F></R>
            <R><F>Jenny Jones</F><F>69 Talkshow Road</F><F>543-210-
               9876</F><F>jjones@tv.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
         </RECORDS>
      </INTERNAL_DATA>
   </DATA>
</PPMLT>
```

A.3 The same dataset, if the Reusable Object occurrences were defined and downloaded earlier

For recurring print projects, a central benefit of PPML is its ability to reference Reusable Object content that was defined earlier using Global scope. This feature allows transmitting smaller datasets and eliminates redundant processing of the reusable content.

The following code shows the PPML Templating dataset that produces the same output as above, presuming that the reusable content was downloaded earlier. The original version was approximately 18k of XML; this version is 10k.

```
<?xml version="1.0" encoding="UTF-8"?>
<PPMLT xmlns="http://www.podi.org/ppmlt/ppmlt001.xsd">
   <TEMPLATE Format="application/xslt+xml">
      <INTERNAL_DATA>
         <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:svg="svg" version="1.0">
            <xsl:output indent="yes"/>
            <xsl:strip-space elements="*"/>
            <xsl:template match="/">
               <!-- Copyright Atlas Software BV -->
```

The start-of-file information is much shorter.

```
            <PPML>
               <DOCUMENT_SET Label="Job Number 1">
                  <IMPOSITION Name="Imporef">
                     <SIGNATURE Nrows="1" Ncols="1">
                        <CELL Row="1" Col="1" Face="Up" PageOrder="s"/>
                     </SIGNATURE>
                  </IMPOSITION>
                  <PRINT_LAYOUT>
                    <PAGE_LAYOUT TrimBox="0 0 612 792"/>
                    <SHEET_LAYOUT Hsize="612" Vsize="792">
                       <IMPOSITION_REF Name="Imporef"/>
                    </SHEET_LAYOUT>
                  </PRINT_LAYOUT>
                  <PRIVATE_INFO Creator="Xeikon" Identifier="MasterVDF"/>
```

The blue-shaded portion, which defines the PPML Document element that will be output for each CUSTOMER *element, is the same as shown above.*

```
<xsl:for-each select="//CUSTOMER">
    <DOCUMENT>
      <PAGE>
        <MARK Position="0 0">
          <OCCURRENCE_REF Ref="XMASTER_BackForeground_1_1"
             Environment="Demo"/>
        </MARK>
        <MARK Position="0 0">
          <OCCURRENCE_REF Ref="XMASTER_ OldsMobile.eps_1 0 0 1 0 0"
             Environment="Demo"/>
        </MARK>
        <MARK Position="334 605">
          <OCCURRENCE_REF Ref="BackForeground_2_1" Environment="Demo"/>
        </MARK>
        <MARK Position="334 605">
          <OBJECT Position="0 0">
             <SOURCE Format="image/svg-xml" Dimensions="165 15">
               <INTERNAL_DATA>
               <svg:svg width="165pt" height="15pt">
                  <svg:text x="82.5pt" y="10pt" font-family="Helvetica" font-
                     size="10pt" word-spacing="1.294pt" letter-spacing=".129pt"
                     text-anchor="middle" fill="rgb(255,255,255)">
                     <xsl:value-of select="NAME"/>
                  </svg:text>
               </svg:svg>
               </INTERNAL_DATA>
             </SOURCE>
          <VIEW>
             <TRANSFORM Matrix="1 0 0 1 0 0"/>
          </VIEW>
        </OBJECT>
        </MARK>
        <MARK Position="334 548.5">
          <OCCURRENCE_REF Ref="BackForeground_3_1" Environment="Demo"/>
        </MARK>
        <MARK Position="334 548.5">
          <OBJECT Position="0 0">
             <SOURCE Format="image/svg-xml " Dimensions="165 15">
               <INTERNAL_DATA>
                  <svg:svg width="165pt" height="15pt">
                     <svg:text x="82.5pt" y="10pt" font-family="Helvetica"
                        font-size="10pt" word-spacing="1.021pt" letter-
                        spacing="0.102pt" text-anchor="middle"
                        fill="rgb(255,255,255)">
                        <xsl:value-of select="STREET"/>
                     </svg:text>
                  </svg:svg>
               </INTERNAL_DATA>
             </SOURCE>
             <VIEW>
               <TRANSFORM Matrix="1 0 0 1 0 0"/>
             </VIEW>
          </OBJECT>
        </MARK>
        <MARK Position="334 494">
          <OCCURRENCE_REF Ref="BackForeground_4_1" Environment="Demo"/>
        </MARK>
        <MARK Position="334 494">
          <OBJECT Position="0 0">
             <SOURCE Format="image/svg-xml " Dimensions="165 14">
                <INTERNAL_DATA>
                   <svg:svg width="165pt" height="14pt">
                      <svg:text x="82.5pt" y="9.5pt" font-family="Helvetica"
                         font-size="10pt" letter-spacing=".341pt" text-
                         anchor="middle" fill="rgb(255,255,255)">
                         <xsl:value-of select="PHONE"/>
```

```
                           </svg:text>
                        </svg:svg>
                     </INTERNAL_DATA>
                  </SOURCE>
                  <VIEW>
                     <TRANSFORM Matrix="1 0 0 1 0 0"/>
                  </VIEW>
               </OBJECT>
            </MARK>
            <MARK Position="334 438">
               <OCCURRENCE_REF Ref="BackForeground_5_1" Environment="Demo"/>
            </MARK>
            <MARK Position="334 438">
               <OBJECT Position="0 0">
                  <SOURCE Format="image/svg-xml " Dimensions="164 15">
                     <INTERNAL_DATA>
                        <svg:svg width="164pt" height="15pt">
                           <svg:text x="82pt" y="9.75pt" font-family="Helvetica"
                              font-size="9pt" letter-spacing="0.05pt" text-
                              anchor="middle" fill="rgb(255,255,255)">
                              <xsl:value-of select="EMAIL"/>
                           </svg:text>
                        </svg:svg>
                     </INTERNAL_DATA>
                  </SOURCE>
                  <VIEW>
                     <TRANSFORM Matrix="1 0 0 1 0 0"/>
                  </VIEW>
               </OBJECT>
            </MARK>
            <MARK Position="84 582.5">
               <OCCURRENCE_REF Ref="BackForeground_6_1" Environment="Demo"/>
            </MARK>
            <MARK Position="84 582.5">
               <OBJECT Position="0 0">
                  <SOURCE Format="image/svg-xml " Dimensions="190 13">
                    <INTERNAL_DATA>
                       <svg:svg width="190pt" height="13pt">
                          <svg:text x="95pt" y="8.5pt" font-family="Helvetica" font-
                             size="8pt" word-spacing="1.789pt" letter-
                             spacing="0.179pt" text-anchor="middle"
                             fill="rgb(255,255,255)">
                             <xsl:value-of select="DESCRIPTION"/>
                          </svg:text>
                       </svg:svg>
                    </INTERNAL_DATA>
                  </SOURCE>
                  <VIEW>
                     <TRANSFORM Matrix="1 0 0 1 0 0"/>
                  </VIEW>
               </OBJECT>
            </MARK>
            <MARK Position="84 598.2">
               <OCCURRENCE_REF Ref="BackForeground_7_1" Environment="Demo"/>
            </MARK>
            <MARK Position="84 598.2">
               <OCCURRENCE_REF Environment="Demo">
                  <xsl:attribute name="Ref">
                  <xsl:value-of select="IMAGE"/>
                  <xsl:text>_1 0 0 1 -0.04066 -0.227</xsl:text>
                  </xsl:attribute>
               </OCCURRENCE_REF>
            </MARK>
          </PAGE>
        </DOCUMENT>
       </xsl:for-each>
       </DOCUMENT_SET>
    </PPML>
 </xsl:template>
</xsl:stylesheet>
```

```
   </INTERNAL_DATA>
</TEMPLATE>
```

The Data Mapper script (shown here in gray) and the Data element are the same as shown above.

```
<DATA_MAPPER Format="application/xslt+xml">
   <INTERNAL_DATA>
      <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
         xmlns:fo="http://www.w3.org/1999/XSL/Format">
        <xsl:output indent="yes"/>
        <xsl:template match="R">
           <CUSTOMER>
              <NAME>
                <xsl:value-of select="F[1]"/>
              </NAME>
              <STREET>
                <xsl:value-of select="F[2]"/>
              </STREET>
              <PHONE>
                <xsl:value-of select="F[3]"/>
              </PHONE>
              <EMAIL>
                <xsl:value-of select="F[4]"/>
              </EMAIL>
              <DESCRIPTION>
                <xsl:value-of select="F[5]"/>
              </DESCRIPTION>
              <IMAGE>
                <xsl:value-of select="F[6]"/>
              </IMAGE>
           </CUSTOMER>
         </xsl:template>
         <xsl:template match="/">
           <CUSTOMERS>
              <xsl:apply-templates/>
           </CUSTOMERS>
        </xsl:template>
      </xsl:stylesheet>
   </INTERNAL_DATA>
</DATA_MAPPER>
<DATA Format="application/xml">
  <INTERNAL_DATA>
      <RECORDS>
         <R><F>Cynthia Proctor</F><F>625 Missouri Street</F><F>510-372-7500</F>
            <F>dcgraphicdesigns@hotmail.com</F><F>1998 Purple Intrigue</F>
            <F>PURPLE</F></R>
         <R><F>Dr. Loose</F><F>Whoville</F><F>123-345-5678</F>
            <F>Loose@whoville.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
         <R><F>Henry Polard</F><F>33 World Trade Blvd.</F><F>650-855-9367</F>
            <F>polard@wenet.net</F><F>1998 Silver Intrigue</F><F>SILVER</F></R>
         <R><F>Al Joshua</F><F>4567 My Way</F><F>123-456-789</F>
            <F>ajoshua@psmail.com</F><F>1998 Silver Intrigue</F><F>SILVER</F></R>
         <R><F>Michelle Walker</F><F>860 36th Ave.</F><F>415/831-1019</F>
            <F>shelwalker@aol.com</F><F>1998 Green/gray Intrigue</F>
            <F>GREEN/GRAY</F></R>
         <R><F>Craig Kohler</F><F>860 36th Ave.</F><F>415/831/1019</F>
            <F>craig.kohler@schwab.com</F><F>1998 Black Intrigue</F><F>BLACK</F></R>
         <R><F>Ken Griffith</F><F>34286 Quartz St.</F><F>510-796-4975</F>
            <F>ken_griffith@splashtech.com</F><F>1998 White Intrigue</F>
            <F>WHITE</F></R>
         <R><F>Harry Raaphorst</F><F>Buys Ballotstraat 17-19</F><F>31-341-426700</F>
            <F>harry.raaphorst@atlasssoftware.nl</F><F>1998 Blue
            Intrigue</F><F>BLUE</F></R>
         <R><F>Michael Barnes</F><F>The Maxwell Company</F><F>415-123-4567</F>
            <F>maxwell@edu</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
         <R><F>Gregg Fox</F><F>200 Canal View Blvd. 831</F><F>716-427-4262</F>
            <F>gregg_fox@mc.xerox.com</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
         <R><F>Paul Lorton, Jr</F><F>1265 Altschul Av.</F><F>650-854-
            2406</F><F>lorton@usfca.edu</F><F>1998 Red Intrigue</F><F>RED</F></R>
             <R><F>Linda Jackson</F><F>405 - 1263 Barclay Street</F><F>604-844-2253</F>
                <F>linda_jackson@splashtech.com</F><F>1998 Black Intrigue</F> <F>BLACK</F>
                </R>
             <R><F>Denis Severson</F><F>3400 Hillview Ave.</F><F>650-813-7158</F>
                <F>severson@parc.xerox.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
             <R><F>Jindong Chen</F><F>3400 Hillview Ave, PAHV 12</F><F>650-813-7338</F>
                <F>jchen@parc.xerox.com</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
             <R><F>Gary Roth</F><F>8758 Wescott Court</F><F>619-484-3226</F>
                <F>gary_roth@splashtech.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
             <R><F>Susan Prischmann</F><F>3930 North Pinegrove, Apt.</F><F>312-849-4361</F>
                <F>sprischmann@currentassets.com</F><F>1998 Green/charcoal
                Intrigue</F><F>GREENCHARCOAL</F></R>
             <R><F>Sue Hoffmann</F><F>2000 Powell Street</F><F>657-1777</F>
                <F>sue_hoffmann@thenet.com</F> <F>1998 Red Intrigue</F><F>RED</F></R>
             <R><F>Rick Placak</F><F>130 So. Center</F><F>702-329-
                3145</F><F>rplacak@thenet.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
             <R><F>Betsy Pryser</F><F>1130 N. Dearborn, #1603</F><F>312-397-
                9250</F><F>epryser@ix.netcom.com</F><F>1998 Silver
                Intrigue</F><F>SILVER</F></R>
             <R><F>Mike Mayo</F><F>124 West Oxmoor</F><F>205-942-
                2222</F><F>jmmayo@worldnet.att.net</F><F>1998 Gold
                Intrigue</F><F>GOLD</F></R>
             <R><F>Armand Petri</F><F>1508 Blackhawk Drive</F><F>408 735
                9482</F><F>apetri@aol.com</F><F>1998 Purple Intrigue</F><F>PURPLE</F></R>
             <R><F>Ted DiSilvestre</F><F>333 W. San Carlos St.</F><F>408-536-
                6508</F><F>tdisilve@adobe.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
             <R><F>Dean Griswold</F><F>6947 West Oak Ct.</F><F>916-725-7739</F>
                <F>griswold@ix.netcom.com</F><F>1998 Green/gray Intrigue</F>
                <F>GREEN/GRAY</F></R>
             <R><F>John Doe</F><F>46 Nowhere Street</F><F>654-321-0987</F>
                <F>john_doe@nowhere.com</F><F>1998 Black Intrigue</F><F>BLACK</F></R>
             <R><F>Jenny Jones</F><F>69 Talkshow Road</F><F>543-210-
                9876</F><F>jjones@tv.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
          </RECORDS>
       </INTERNAL_DATA>
    </DATA>
 </PPMLT>
```

A.4 Leanest form: Template, Reusable Content, and Data Mapper have all been downloaded in advance

In this version the XML code is reduced to 3.8k – essentially the size of the variable data itself. The TEMPLATE and DATA_MAPPER elements each contain an EXTERNAL_DATA reference to a previously defined file that was downloaded earlier. Similar results could have been achieved using TEMPLATE_REF and DATA_MAPPER_REF.

```
<?xml version="1.0" encoding="UTF-8"?>
<PPMLT xmlns="http://www.podi.org/ppmlt/ppmlt001.xsd">
   <TEMPLATE Format="application/xslt+xml">
      <EXTERNAL_DATA Src="Project.xsl"/>
   </TEMPLATE>
   <DATA_MAPPER Format="application/xslt+xml">
      <EXTERNAL_DATA  Src="MyMapper.xsl"/>
   </DATA_MAPPER>
   <DATA Format="application/xml">
     <INTERNAL_DATA>
         <RECORDS>
            <R><F>Cynthia Proctor</F><F>625 Missouri Street</F><F>510-372-7500</F>
               <F>dcgraphicdesigns@hotmail.com</F><F>1998 Purple Intrigue</F>
               <F>PURPLE</F></R>
            <R><F>Dr. Loose</F><F>Whoville</F><F>123-345-5678</F>
               <F>Loose@whoville.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
            <R><F>Henry Polard</F><F>33 World Trade Blvd.</F><F>650-855-9367</F>
               <F>polard@wenet.net</F><F>1998 Silver Intrigue</F><F>SILVER</F></R>
            <R><F>Al Joshua</F><F>4567 My Way</F><F>123-456-789</F>
               <F>ajoshua@psmail.com</F><F>1998 Silver Intrigue</F><F>SILVER</F></R>
            <R><F>Michelle Walker</F><F>860 36th Ave.</F><F>415/831-1019</F>
               <F>shelwalker@aol.com</F><F>1998 Green/gray Intrigue</F>
               <F>GREEN/GRAY</F></R>
            <R><F>Craig Kohler</F><F>860 36th Ave.</F><F>415/831/1019</F>
               <F>craig.kohler@schwab.com</F><F>1998 Black Intrigue</F><F>BLACK</F></R>
            <R><F>Ken Griffith</F><F>34286 Quartz St.</F><F>510-796-4975</F>
               <F>ken_griffith@splashtech.com</F><F>1998 White Intrigue</F>
               <F>WHITE</F></R>
            <R><F>Harry Raaphorst</F><F>Buys Ballotstraat 17-19</F><F>31-341-426700</F>
               <F>harry.raaphorst@atlasssoftware.nl</F><F>1998 Blue
               Intrigue</F><F>BLUE</F></R>
            <R><F>Michael Barnes</F><F>The Maxwell Company</F><F>415-123-4567</F>
               <F>maxwell@edu</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
            <R><F>Gregg Fox</F><F>200 Canal View Blvd. 831</F><F>716-427-4262</F>
               <F>gregg_fox@mc.xerox.com</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
            <R><F>Paul Lorton, Jr</F><F>1265 Altschul Av.</F><F>650-854-
               2406</F><F>lorton@usfca.edu</F><F>1998 Red Intrigue</F><F>RED</F></R>
            <R><F>Linda Jackson</F><F>405 - 1263 Barclay Street</F><F>604-844-2253</F>
               <F>linda_jackson@splashtech.com</F><F>1998 Black Intrigue</F> <F>BLACK</F>
               </R>
            <R><F>Denis Severson</F><F>3400 Hillview Ave.</F><F>650-813-7158</F>
               <F>severson@parc.xerox.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
            <R><F>Jindong Chen</F><F>3400 Hillview Ave, PAHV 12</F><F>650-813-7338</F>
               <F>jchen@parc.xerox.com</F><F>1998 Gold Intrigue</F><F>GOLD</F></R>
            <R><F>Gary Roth</F><F>8758 Wescott Court</F><F>619-484-3226</F>
               <F>gary_roth@splashtech.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
            <R><F>Susan Prischmann</F><F>3930 North Pinegrove, Apt.</F><F>312-849-4361</F>
               <F>sprischmann@currentassets.com</F><F>1998 Green/charcoal
               Intrigue</F><F>GREENCHARCOAL</F></R>
            <R><F>Sue Hoffmann</F><F>2000 Powell Street</F><F>657-1777</F>
               <F>sue_hoffmann@thenet.com</F> <F>1998 Red Intrigue</F><F>RED</F></R>
            <R><F>Rick Placak</F><F>130 So. Center</F><F>702-329-
               3145</F><F>rplacak@thenet.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
            <R><F>Betsy Pryser</F><F>1130 N. Dearborn, #1603</F><F>312-397-
               9250</F><F>epryser@ix.netcom.com</F><F>1998 Silver
               Intrigue</F><F>SILVER</F></R>
            <R><F>Mike Mayo</F><F>124 West Oxmoor</F><F>205-942-
               2222</F><F>jmmayo@worldnet.att.net</F><F>1998 Gold
               Intrigue</F><F>GOLD</F></R>
            <R><F>Armand Petri</F><F>1508 Blackhawk Drive</F><F>408 735
               9482</F><F>apetri@aol.com</F><F>1998 Purple Intrigue</F><F>PURPLE</F></R>
            <R><F>Ted DiSilvestre</F><F>333 W. San Carlos St.</F><F>408-536-
               6508</F><F>tdisilve@adobe.com</F><F>1998 Blue Intrigue</F><F>BLUE</F></R>
            <R><F>Dean Griswold</F><F>6947 West Oak Ct.</F><F>916-725-7739</F>
               <F>griswold@ix.netcom.com</F><F>1998 Green/gray Intrigue</F>
               <F>GREEN/GRAY</F></R>
            <R><F>John Doe</F><F>46 Nowhere Street</F><F>654-321-0987</F>
               <F>john_doe@nowhere.com</F><F>1998 Black Intrigue</F><F>BLACK</F></R>
            <R><F>Jenny Jones</F><F>69 Talkshow Road</F><F>543-210-
               9876</F><F>jjones@tv.com</F><F>1998 Red Intrigue</F><F>RED</F></R>
         </RECORDS>
      </INTERNAL_DATA>
   </DATA>
</PPMLT>
```

A.5 Example results

The table below summarizes the effect of the templating application shown in the example above. In this example, a single PPML Document requires approximately 4.5k bytes of PPML code, and the Reusable Object definitions require approximately 8k bytes. Each record of customer data is about 140 bytes.

A.5.1 PPML without templating

	Total data required to produce this many Instance Documents			
Method	**25 Instance Documents**	**1,000 Instance Documents**	**10,000 Inst. Documents**	**100,000 Inst. Documents**
PPML file size including Reusable Object definitions ("ROs")	ROs: 8k PPML excluding Inst. Docs: 10k Inst. Docs: 122k **Total: 140k**	ROs: 8k PPML excluding Inst. Docs: 10k Inst. Docs: 4.5MB **Total: 4.52MB**	ROs: 8k PPML excluding Inst. Docs: 10k Inst. Docs: 45MB **Total: 45MB**	ROs: 8k PPML excluding Inst. Docs: 10k Inst. Docs: 450MB **Total: 450MB**
PPML file size with ROs pre-loaded	PPML excluding Inst. Docs: 10k Inst. Docs: 122k **Total: 132k**	PPML excluding Inst. Docs: 10k Inst. Docs: 4.5MB **Total: 4.51MB**	PPML excluding Inst. Docs: 10k Inst. Docs: 45MB **Total: 45MB**	PPML excluding Inst. Docs: 10k Inst. Docs: 450MB **Total: 450MB**

A.5.2 With templating

	Data required to produce this many Instance Documents			
Method	**25 Instance Documents**	**1,000 Instance Documents**	**10,000 Inst. Documents**	**100,000 Inst. Documents**
PPMLT file size Including Reusable Object definitions ("ROs")	Data: 3.5k ROs: 8k Template & Mapper: 6.5k **Total: 18k**	Data: 140k ROs: 8k Template & Mapper: 6.5k **Total: 154k**	Data: 1.4MB ROs: 8k Template & Mapper: 6.5k **Total: 1.41MB**	Data: 14MB ROs: 8k Template & Mapper: 6.5k **Total: 14MB**
PPMLT file size with ROs pre-loaded	Data: 3.5k Template & Mapper: 6.5k **Total: 10k**	Data: 140k Template & Mapper: 6.5k **Total: 146k**	Data: 1.4MB Template & Mapper: 6.5k **Total: 1.41MB**	Data: 14MB Template & Mapper: 6.5k **Total: 14MB**
PPMLT file size with ROs, template, and mapper all pre-loaded	Data: 3.5k Other: 0.3k **Total: 3.8k**	Data: 140k Other: 0.3k **Total: 143k**	Data: 1.4MB Other: 0.3k **Total: 1.4MB**	Data: 14MB Other: 0.3k **Total: 14MB**

Exhibit 3
to Gauthier Declaration

Speaking In Tongues: Sorting Out Variable Data Printing Languages

By Eliot Harper

As the digital color print market continues to grow, adoption of variable data printing is increasing rapidly. Personalization has become commonplace. In this overview, we look at the proliferation of digital file formats devised to cope with the practical side of variable print communication.

Variable data printing has long since been opened out of it's flat-pack carton and assembled to join the rest of the industry acronym furniture, along with UCR, GCR, CtP, JDF, CIP4 and a few forgotten armchairs. Today, VDP accounts for a healthy share of print volume; 37% of graphic arts firms (printers and trade shops) produce some sort of VDP jobs in-house, up from 28% one year ago*. The variety of VDP applications in use today ranges from simple business correspondence with name, address and basic information changing for each recipient, through to direct mail applications where graphical and text elements are switched based on a set of business rules to produce a unique composition, customized to each recipient.

Furthermore, VDP is no longer limited to print. The term has been adopted as "variable data publishing" to include other media channels. A multichannel marketing campaign can incorporate personalized content in a printed piece and offer supporting content and response channels through Web, e-mail and mobile devices.

Although VDP has been used for over a decade, personalized digital printing is hardly new. Essential mail (bills, statements, etc.) has been produced on digital printers ever since Xerox introduced the first 9700 laser printer in 1977. These essential documents were printed from mainframe environments using optimized data stream languages such as Metacode and LCDS, which are still widely used in transactional printing environments.

When digital color printing emerged in the early 1990s, these legacy data stream languages were not suitable for personalized color documents, and as a result many printer and RIP vendors developed their own variable information (VI) languages.

Today, VDP software products and RIPs support a host of different VI languages, and selecting an appropriate language can be a somewhat daunting task. In this article we identify and describe all of the variable information languages so that you can make an informed decision when you choose an appropriate language for your VDP work.

Most operating system print drivers create a page description language, or PDL (typically PostScript), by processing each page of a document individually. If such print drivers are used to create a print-ready file for VDP applications, the resulting files would contain a separate page (or pages) for each record. This can result in very large file sizes and might require a considerable amount of time for RIPs or print controllers to interpret and process the data.

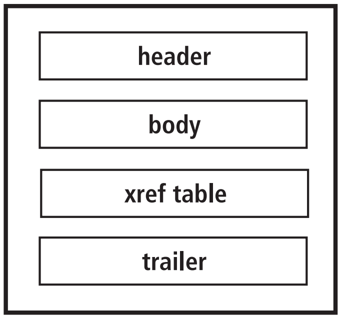
To address this issue, several variable information languages have been developed specifically to contain page description information for VDP applications that overcome the limitations of traditional page description languages. These VI languages enable print files to be created using various optimization techniques, including object caching and custom page instructions that can be interpreted by supported RIPs.

Optimized Portable Document Format (PDF)

Adobe Systems' portable document format (PDF), developed in 1993, represents documents within a device- and display resolution-independent fixed-layout document format. Built on a subset of PostScript, PDF files encapsulate a complete description of all objects within a document, including text, fonts, graphics and vector objects. PDF files also include a structured storage system to bundle all page objects and other content into a single file, using data compression where appropriate.

The general structure of a PDF file consists of a header, body, cross-reference (xref) table and trailer. The trailer contains pointers to the xref table and to key objects contained in the trailer dictionary. The xref table contains pointers to all the objects included in the PDF file. It identifies how many objects are in the table, where the object begins (the offset) and its length in bytes. The body contains all the object information: fonts, images, text and other object types.

Since PDF was developed as a fixed-layout document format, a PDF file



contains separate pages for each page to be printed. As a result, when used for variable data printing applications, the PDF file will contain a separate page (or pages) for each record, which can result in a PDF file several thousand pages long. However, PDF enables file optimization through its ability to reuse common objects in the document — both text and images. This not only results in a smaller file size, but also enables faster file processing at the RIP, as reused objects are cached at the RIP. This type of file optimization is commonly referred to as "thin" or "optimized" PDF.

As a result, the difference in file size between a PDF with 100 records and the same application with 1,000 records could be fairly minor, as common elements are stored and reused across multiple pages.

However, an optimized PDF file cannot be created by just any PDF driver, since the driver needs to identify repeating objects in a file and format them as reused content. One common method for creating an optimized PDF file is to use Adobe's PDF driver (installed with Acrobat) or create an optimized PostScript file, then create a PDF file from the PostScript file using Adobe Acrobat Distiller.

Optimized PostScript

Adobe Systems developed PostScript in 1984 for the desktop publishing market. PostScript and PDF share many similarities, as both file formats describe text and graphics. The main difference is that PostScript is a page description language and also a programming language that is processed by an interpreter to generate an image, whereas PDF is a file format and not a programming language.

As noted earlier, operating system print drivers create a PostScript file by writing each page individually, including all of the objects on every page. Therefore, creating a VDP application for many records can result in an extremely large file. This typically occurs when creating VDP applications using entry-level data merge utilities, as they rely on the operating system driver to create the print file, such as the built-in data merge functionality in Microsoft Word or Adobe InDesign. This type of PostScript file is often referred to as "fat" PostScript.

Similar to the PDF imaging model, PostScript also supports reusable content (or "form caching") of repeating objects by using the PostScript Level 2 form caching environment. This enables the creation of "thin" or "optimized" PostScript files, where repeating objects are only included once in the document. However, the software or driver used to create the PostScript file must support form caching to take advantage of this type of optimization.

Because PostScript is a programming language, PostScript files can include programmatic commands to draw page objects from data. For example, pie charts, line graphs or bar charts can be drawn from included data. Text content can also be written programmatically in PostScript so that words or blocks of text can change depending on defined rules. Since these commands are instructions in the PostScript file, pages can be composed by the PostScript interpreter (the raster image processor, or RIP) instead of using VDP software products for file composition.

Output device commands can also be included as variable instructions in PostScript. This enables the PostScript interpreter to call supported page tray and finishing features such as stapling, collating and folding. A programmed business rule could be defined in the PostScript file to delineate page tray-pull instructions for each record based on a value in the data. Letters for "regular" members might be printed on plain paper and letters for "premium" members on high-quality paper, for example.

The difference in file size between a PDF with 100 records and the same application with 1,000 records could be fairly minor, as common elements are stored and reused across multiple pages.

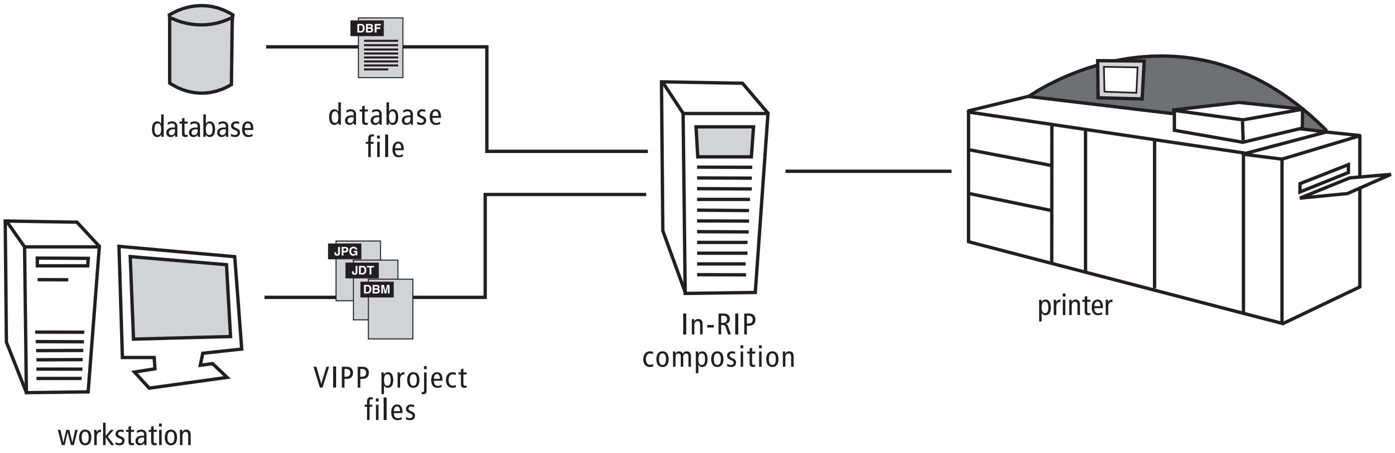
Although many VDP software products offer "optimized PostScript" output, only a few support RIP-level composition or "dynamic document composition." Most VDP software products use their own composition engine or page layout software (Adobe InDesign or QuarkXPress). This is not necessarily a disadvantage, as PostScript does not have the same level of typographic and graphic control as page layout software. However, RIP-side composition can present significant performance benefits since documents are composed and rasterized at the same time instead of a two-stage process in which documents are composed (as a PostScript file) on a desktop or server before being rasterized at the RIP.

Printer Command Language (PCL)

Printer Command Language (PCL) is a printer protocol originally developed by Hewlett Packard in the 1980s for early inkjet printers. PCL has been released in varying levels over the past 20 years and is now supported across a range of digital printing technologies.

PCL levels 1 through 5e/5c are command-based languages using control sequences that are processed and interpreted in the order in which they are received. In 1995, HP introduced PCL 6, a language very different from earlier PCL versions because it provided a stack-based, object-oriented protocol similar to PostScript.

Although PCL is not a variable information language, its file structure can store common page elements, so repeating page objects (such as images) only need to be stored in the PCL file once. PCL is supported by a few VDP software products.



Variable-data Intelligent PostScript Printware (VIPP)

Developed by Xerox, VIPP is a PostScript-based language designed to take advantage of the powerful programming features of PostScript, as well as address the limitations of PostScript in VDP applications.

With VIPP commands, VDP applications can remain independent of PostScript, since VIPP can use higher-level PostScript operators. They provide support for common VDP application requirements, including data-driven graphics and text commands for text highlighting and reflow across multiple frames, including pages.

VIPP can also process native data streams from legacy (line data) to XML, enabling independent data production and VDP application design. As a result, VIPP application resource files can be packaged as VIPP project container (VPC) files and loaded on the RIP, where the raw data file can be sent to the RIP to trigger document composition and production.

Although many VDP software products offer support for VIPP, only a few products take advantage of VIPP's intepreter-level composition model and "just send the data" workflow. Most variable data printing software products use their own composition engine or page layout software to compose the document and only use a few VIPP commands (such as form object caching). As indicated earlier, this is not necessarily a disadvantage, since page layout software such as InDesign includes powerful page layout features, drawing tools, typography control and more. As a result, document composition using page layout software might be more suited for design-intensive applications.

Variable Print Specification (VPS)

VPS is a PostScript-based language developed by Scitex (now Print On-Demand Solutions, a Kodak company). The VPS imaging model is constructed of pages, and each page is constructed of elements. There are two types of elements: reusable and non-reusable or "inline" elements. In this respect VPS is similar to other VI languages that use reusable element models, such as PDF, PostScript, PPML and VIPP. VPS is supported across a number of Creo/Kodak and EFI OEMed RIPs.

Personalized Print Markup Language (PPML)

PPML is an XML-based variable information language defined and developed by the Digital Print Initiative (PODi), a not-for-profit industry consortium of vendor companies that fosters digital printing growth through market and standards development activities. The PPML framework is built on two core methods, object-level granularity and reusable content.

Object-level granularity describes content objects on one page instead of individual pages. Reusable content refers to the ability to temporarily or permanently save page content to the RIP memory and use it throughout the composition of the VDP document or other VDP documents.

Reusable content can include fonts, graphics, images and other digital assets. Similar to VIPP and VPS, PPML content can reside locally on the RIP or it can be retrieved from another device by "referencing" remote resources via URLs, eliminating the need to send all the resources with the print job.

As PPML is an XML- (text) based language, it can't contain binary data. As a result, all internal graphical content in the PPML file (such as non-referenced content) has to be encoded. This encoding can result in significantly larger file sizes than comparable languages (like VPS or VIPP). As a result, it is good practice to use referenced content when creating PPML.

Like several other VI languages, PPML can dynamically merge objects (text and images) on a template at the RIP from the supplied dataset and digital assets.

Due to the large number of elements within the PPML definition and varying levels of PPML implementation across vendors, a Graphic Arts Conformance Specification has been created to define required PPML elements to ensure interoperability across different VDP software and RIPs. This specification defines how device-dependent colors are handled, which digital asset formats are supported and other related requirements.

Personalized Print Markup Language/Variable Data Exchange (PPML/VDX)

The American National Standards Institute (ANSI, **www.ansi.org**) approved the PPML/VDX standard, developed by the Committee for Graphic Arts Technologies Standards (CGATS), early in 2002. Formerly known as VDX, PPML/VDX is based on a subset of the PPML specification.

A PDF-based standard, PPML/VDX uses a subset of PPML to define the reusable content within the PDF file. A PPML/VDX file can consist of one or more files. A collection of one or more files is referred to as a "PPML/VDX Instance." In its most basic form, an instance will always contain a PPML/VDX layout file.

The layout file is a PDF file that functions as a container for the VDP template and variable elements (text and images). Although the layout file is a PDF, it uses the filename extension .vdx to signify that it is not a regular PDF file.

FreeForm 2, an extension of FreeForm, offers all the functionality of FreeForm while also providing full support for page picking and greater flexibility for database integration.

The layout file includes PPML information that defines the layout of the document, the structure of pages and the variable elements (text and images) used in the VDP document and how elements are joined to pages, while the PDF pages contain the variable elements.

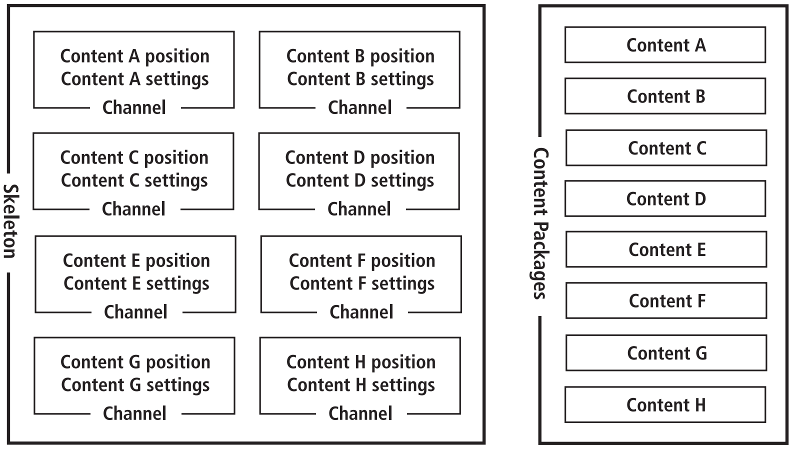
The layout file can also include job definition format (JDF) job ticketing instructions. The PPML information, variable elements and JDF instructions can either be contained within the layout file or referenced from the layout file and organized as individual files within the instance.

The layout file also contains a content table, or "checklist," that can verify that all required elements for the VDP application have been included in the instance.

FreeForm

Developed by Electronics for Imaging (EFI), FreeForm was one of the first color VI languages in the industry and is standard on most of EFI's Fiery branded RIPs.

The FreeForm imaging model uses two layers. A master layer (or template file) contains all static data (static images and PDF pages), while the variable layer contains all variable data (variable images and text). The layers, or files, are sent to the RIP separately. The template file is loaded onto the FreeForm-enabled RIP, where it is rasterized and assigned a user-defined numeric ID.



Once the template file is loaded on the RIP, the variable data layer (or job) is referenced to the corresponding template ID and is sent to the RIP. The RIP rasterizes the variable data job and overlays the rasterized master data with the rasterized data of the variable data job.

Using this template-based approach to VDP, the same master template can be reused for different or versioned VDP documents. In addition, FreeForm supports image caching, where repeating variable images are stored in reusable forms within the variable data layer.

FreeForm is an ideal variable information language for entry-level VDP documents and applications, but it offers only basic VI functionality and has limited support for advanced VI commands such as page picking (the ability to define individual pages in a document) and conditionally skipping layouts.

FreeForm 2

FreeForm 2, an extension of FreeForm, offers all the functionality of FreeForm while also providing full support for page picking and greater flexibility for database integration.

In addition, the master "template ID" in FreeForm 2 consists of a user-defined name rather than a number, which enables easier template management.

Job Layout (JLT)

Developed by Indigo (now HP), the job layout (JLT) language is a proprietary job description language used by HP Indigo Digital Presses.

JLT is not just a VI language, it is also a file format that defines document job structure and includes basic job ticket information. This proprietary file format enables integration of the print job with the software and hardware architecture of the HP Indigo press.

A JLT file contains two parts, a "skeleton" consisting of channels and "content packages." Channels in the skeleton define the position of the content (page objects) and settings that define the transformation of each object — for example image scaling and rotation. For variable data printing documents, these channels also define the link to variable content.

Content packages include any static content in the file (text and image page objects, for instance). Variable content is not included in content packages but is loaded on the RIP separately. However, using an optional "rich mode" setting, variable objects can be included as a PostScript file.

HP Indigo requires this unique workflow of separating the file structure from the content, as all page objects (static and variable) need to be rasterized to an internal format (Indigo Compressed Format). The ICF objects in the document are then assembled at the RIP according to the JLT structure or database file (for VDP documents).

One of the core differences in IJPDS compared to other VI languages is its support for built-in parallel processing of a single file.

Variable Data File (VDF)

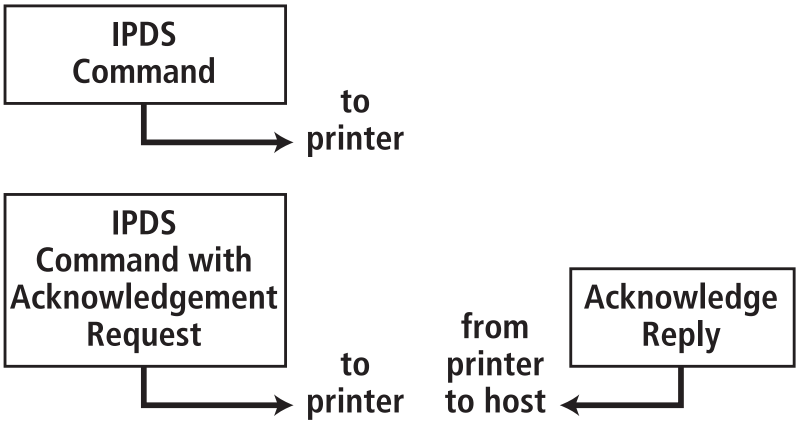
Originally developed by Agfa for ChromaPress, the variable data file format was intended for the Agfa IntelliStream (now Xeikon) digital front end.

Variable data file is a PostScript-based language and the VDF workflow uses a document template that is saved as a PostScript file. Variable page objects are then saved as individual variable data files and a separate VDF file is created for each variable object. In addition, setup (STP) files can be created that contain information about each variable object, with a separate STP file created for each object. When STP files are available, the IntelliStream DFE can use its IntelliCache feature to cache variable objects.

Xeikon replaced the IntelliStream DFE (digital front end) in 2004 with a newer DFE architecture, the X-800, to support the current models of Xeikon print engines. In its new DFE, Xeikon has moved away from VDF support to PPML and PPML/VDX.

Intelligent Printer Data Streams (IPDS)

Developed by IBM for mainframe printing environments, intelligent printer data stream (IPDS) is part of IBM's advanced function presentation (AFP) architecture. The IPDS language contains the information necessary to identify, monitor and control the functions of certain kinds of printers that are used in mainframe environments. This information includes the characteristics of the printer, its resolution, what resources it has, whether it has sufficient memory and whether it can receive and print a job.



The IPDS architecture enables both spooled data (such as fonts, text, images) and print job management controls (like resolution, paper tray handling, media jams) to flow bidirectionally between both the print server (or print driver) and the printer controller. IPDS data streams are only used to carry print instructions and data from the print server to the printer in structured fields. The print controller processes IPDS commands, then returns an acknowledgment back to the print server.

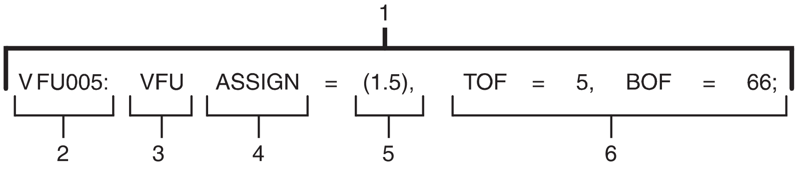
IPDS also provides support for media finishing using printer-attached devices or preprocessing and postprocessing devices. In addition to traditional printer-controlled finishing, constructs are also provided to enable IPDS data streams to be used within universal printer pre- and post-processing interface (UP3I) environments.

Inkjet Printer Data Stream (IJPDS)

Developed by Scitex, IJPDS is a proprietary file format for Scitex (now Kodak) Versamark printers built on a simple binary file format consisting of block-formatted page components.

IJPDS was developed primarily for printing variable text. Fonts are defined in the header file as bitmaps and page layout is based on lines of text. IJPDS support for images is limited and only images that are placed in line with text are supported. Furthermore, images have to be defined as a bitmap font glyph. For full-color images, four separate bitmap images are required (C, M, Y, K), which are placed over one another.

One of the core differences in IJPDS compared to other VI languages is its support for built-in parallel processing of a single file. For example, the IJPDS file could include instructions to process records across eight interpreters (or CPUs). These instructions are defined in the IJPDS file by a job control record that



1. Command statement
2. Identifier
3. Command keyword
4. Parameter keyword
5. Parameter option
6. Additional parameter keywords and options

indicates which interpreter the following records will refer to. The disadvantage of this parallel-processing approach is that the file becomes RIP-dependent, so it isn't possible to send a IJPDS file created for eight interpreters to a different RIP with two interpreters.

Line Conditioned Data Stream (LCDS)

Xerox developed LCDS to allow simple line data to take advantage of Xerox's 9700 and 4000 series printers. LCDS is a set of printing system commands that defines printer properties such as the appearance, output destination and paper feed source of a print job.

LCDS allowed easy migration from earlier impact-based printers by enabling page composition to be performed on the printer controller, where enabled forms, fonts and images were stored directly on the printer controller.

Printing system commands are entered together in a job source library (JSL) file. The JSL file is then compiled as an object file called a job descriptor library (JDL) file that the printing system can read. The printing system then responds to the commands contained in the JDL file and prints the job as it is defined to appear. LCDS does not support color printing.

Metacode

Developed by Xerox, Metacode is a machine code variant of LCDS used to describe text and graphics. Like LCDS, it was developed as a low-level control language for Xerox 9700 and 4000 series printers. It provides greater flexibility than LCDS through its own proprietary metalanguage.

The Metacode language uses hexidecimal character codes to describe the position of data (text and graphics) on a page. This code can also be used for page orientation control and font selection and can enable highlight color support.

Summary

To a large extent, your production environment will govern your choice of variable information languages. Your RIP will determine which VI languages you can support and your data environment will likely dictate which languages you can use (particularly in legacy LCDS, IPDS or Metacode data-stream environments). Whenever you do have a choice, the vast assortment of VI languages available today can make selecting an appropriate language a difficult task.

VI language support varies across different VDP software and RIPs. In creating a VI file, the supported features and composition performance of one VDP software product might be different from another VDP software product. For example, a VIPP project created from XMPie uDirect software will differ from one created by Lytrod Designer software. XMPie uDirect uses InDesign for document design and composition, whereas Lytrod Designer uses its own design environment and packages the application resource files together, and the document is composed at the RIP. This RIP-side composition can offer significant performance benefits over desktop or server-side composition, as the document is composed and interpreted simultaneously. Text, images and data-driven graphics such as pie chart or line graphs are composed by the PostScript interpreter. Desktop or server-side composition software needs to first generate an output file (using page layout software such as InDesign or their own composition engine) before it can be interpreted by the RIP.

The open standards discussion does not really have merit in VDP; if a proprietary file format can offer greater performance and VI feature support than a standards-based format, is the standards-based format better?

Furthermore, VI performance will vary across different RIPs. For example the processing time of a PPML file for one RIP could differ considerably when processed on a RIP by a different vendor, even if the RIPs are running on similar hardware. Different vendors use different approaches to interpreting and rendering variable information, and like with performance-enhancement drugs, "results may vary."

When possible, you should run some benchmark tests on your RIP using selected VDP software products (trial versions are available for most of them) to gauge performance differences among different software, VI languages and application types.

When selecting a VI language, base your choice on your own research and experience and not on market direction or opinions. One popular objection to vendor file formats is that they are regarded as proprietary technologies, not open standards. The open standards discussion does not really have merit in VDP; if a proprietary file format can offer greater performance and VI feature support than a standards-based format, is the standards-based format better?

Another consideration is performance, but this does not need to be the deciding factor. Certain VI formats can be interpreted by RIPs at a rate of several thousands of pages per minute, but can your printer print at thousands of pages per minute? The key to performance is ensuring that you can run your print engine at rated-speed.

Choosing a VI language is a little like buying a new pair of shoes; there are many varieties, each with its own style and benefits, and one size does not fit all. You need to find the pair that fits you. **TSR**

* source: "*Variable Data Printing 2006: Growth and Changes in the Marketplace*" TrendWatch Report

Eliot Harper is workflow marketing manager at Fuji Xerox Australia. He can be contacted at eliot.harper@aus.fujixerox.com.

Exhibit 4
to Gauthier Declaration

HIGH-PERFORMANCE VARIABLE DATA PRINTING USING PDF

By Martin Bailey, Chief Technology Officer, Global Graphics Software.
The UK primary expert on ISO (International Standards Organization) for PDF, PDF/VT and PDF/A.

Introduction

InfoTrends' End-User Workflow Survey, 2010 asked the question "Please select the top two optimized print output formats used for variable data job production". The data that they collated clearly shows that the run-away winner at the top of the list was "Optimized PDF" with nearly 60%.

For years many variable data print (VDP) vendors have said that you can only achieve high throughput on press by using specialist VDP languages; the market appears to disagree.

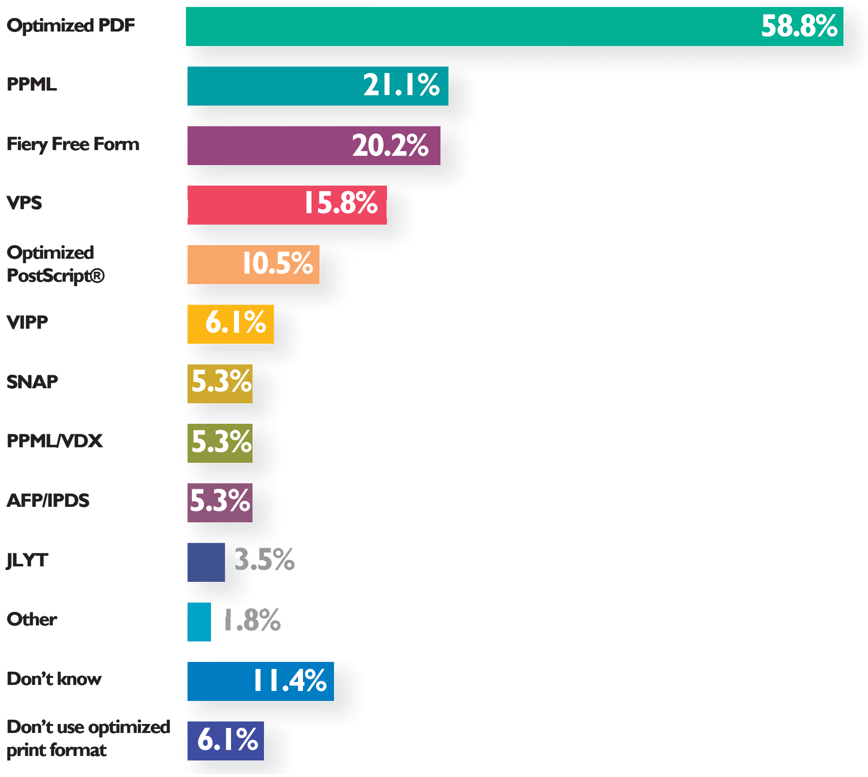
Variable data is now printed at more print sites than ever before, driven by an overall growth in digital printing, and by a transfer from printing customer mail in the data center to workflows that are more closely related to the graphic arts.

> VDP optimizations in Harlequin RIPs provide world beating performance without losing workflow benefits such as viewing.

Optimized print output language usage

Users were asked to select the top two optimized print output formats used for variable data job production (Multiple responses permitted).

End-User Workflow Survey, InfoTrends 2010.



Digital production presses and variable data print have developed greatly over the last decade or so. Presses are much faster than they were ten years ago and often running at higher resolution. The computing power available for inclusion in a controller or digital front end (DFE) has also been increasing, while its cost has dropped. On balance it's now easier to render jobs fast enough to achieve full engine speed on a sheet-fed press than it used to be… as long as you print the simple VDP pages that were being processed back then. A third trend that's occurred at the same time is that the complexity of print jobs has risen, increasing the demands on processing power in the DFE again.

In parallel with that a new breed of ultra high speed ink-jet web press, printing at over 500ft/min (150m/min) has emerged. The Hewlett-Packard T300, T350 and T400 presses are examples of this class of press. Achieving ROI on these requires that they be driven at or near full engine speed, for all of every shift, only stopping for scheduled maintenance.

Traditional VDP formats

A successful personalized marketing campaign needs the printed product to be novel, attractive and compelling enough to persuade the recipient to read it before discarding it. The tools used by designers for creating general and publication print have become richer and more complex over time; designers for VDP pieces (quite naturally) want to take advantage of those tools. This can lead to a tension between designers and the print production team over what features can be used while still achieving high enough performance in the DFE and on press to be commercially viable.

< Presses are much faster than they were ten years ago, often running at higher resolution. Computing power has increased but print jobs are much more complex. >

Vendors have always tried to build solutions that are capable of the most efficient processing possible using technology available at the time, which lead to the creation of a variety of specialist VDP page description languages (PDLs). By using something like PPML it was possible to reduce the amount of processing that the DFE had to do in order to achieve a given final appearance. The tools that create the PPML stream do some of the work for the DFE in identifying which parts of each page are used many times, so the DFE only needs to render each of those shared page elements once. It then renders all of the elements that were not shared. Finally the shared and variable elements for each page are stitched together (often using hardware assistance) and the page is printed.

That model may enable the highest possible throughput in the DFE and the press for relatively simple jobs, but it carries a number of hidden costs:

a) There are many VDP-specific PDLs, some only supported by a single DFE vendor. A print site running presses from multiple suppliers may need to make files differently for each press, leading to higher costs for creation tools and training and a lack of flexibility in late decisions. Even nominally 'universal' VDP PDLs like PPML suffer because it's often implemented differently by each vendor.

b) Several proprietary VDP PDLs include assumptions that all DFEs that will process them include specialist hardware designed to aggregate rasters post RIP. This makes it difficult to scale the use of exactly the same VDP PDL over a whole range of digital presses from light production to high-volume, again meaning that different PDLs are required for different printers and presses.

c) Most VDP-specific PDLs were designed by a vendor who supplies a creation tool or a digital press with its associated DFE, so other aspects of the VDP production process are often not well served by the design; there's more to workflow than making a VDP data stream in one place and printing it through a DFE and press at another, including viewing, proofing, preflight etc.

d) When most of the VDP-specific PDLs were first specified it was possible to use them to create pages as rich as those used in commercial and publication print at the time. Since then the use of live transparency in PDF has become commonplace. PPML has now been updated to v3.0 to address this, but most of the proprietary VDP PDLs have not and PPML 3.0 has not been widely implemented. It's also remained true to its roots in constraining users to the graphical effects that can be processed most efficiently in today's DFEs. By drupa 2012 it's likely to be starting to be seen as overly restrictive as the next generations of DFEs for formats such as optimized PDF deliver higher performance without those limitations.

e) Almost all long VDP jobs are created using specialist tools. But shorter VDP jobs created in-house by companies who have less frequent needs are often made with tools that were not designed to make VDP-specific PDLs. The PSP or CRD still needs to receive the documents to be printed in a stable, reliable format.

It's not all that surprising that a lot of companies creating VDP jobs, and print companies who print them have elected to use PDF instead of something more specialized to the task. The ability to explain to all customers what they need to submit, to send the same file to (almost) all DFEs, to view the final file virtually anywhere, and to create files as rich as the customer demands all go at least some way to balancing out the potential for a drop in performance in the DFE.

< Harlequin VariData ensures that performance can be maximised for VDP jobs created today and into the future. >

VDP in Harlequin RIPs

Global Graphics Software is the creator of the Harlequin RIP, an important component in digital production DFEs. In 2007 it set about ensuring that using PDF for VDP would achieve the highest possible performance. Harlequin VariData™ is an expanded and improved replacement for the PDF Retained Raster functionality, released in Global Graphics' Harlequin Server RIP® version 8.0, launched at drupa 2008.

Harlequin VariData automatically analyses a PDF file to identify those pages that use shared elements. It therefore takes advantage of optimized structures in PDF files made with specialist VDP creation tools, including those saved as PDF/VT (ISO 16612-2). At the same time it works almost as well for PDF files made by general tools that are not specialized for VDP.

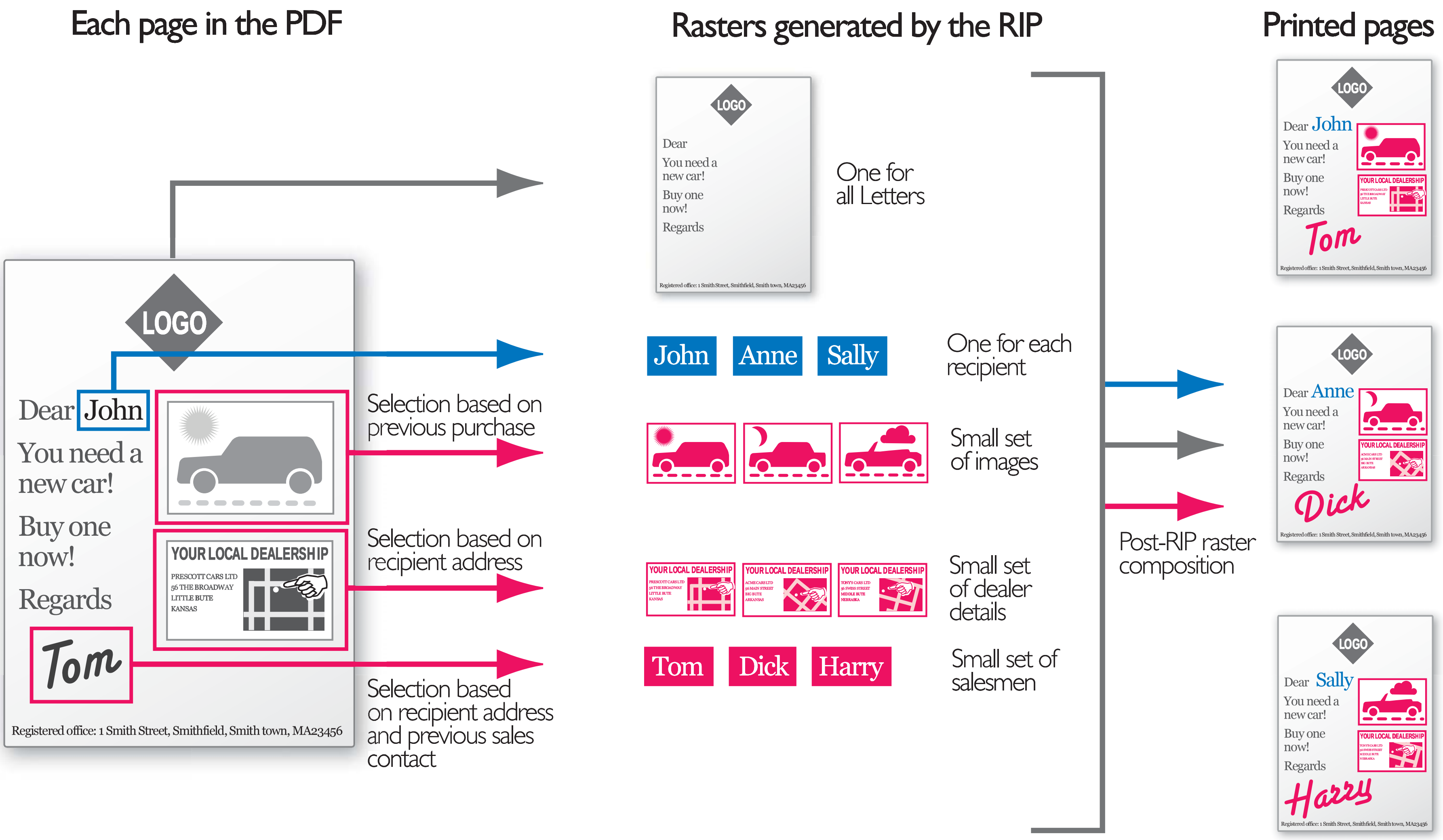
Once a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately. The benefits of the specialized VDP PDLs can therefore be achieved while using PDF.

Harlequin VariData can be used in two configurations:

4

Simplified example of variable data print workflow within a DFE using External Harlequin VariData™ in a Harlequin RIP

Harlequin VariData in the Harlequin RIP automatically recognizes combinations of graphical elements that are re-used on multiple pages in a VDP job, and delivers each as a separate raster. Post-RIP raster composition technology can be used to build each final page for print, producing maximum throughput for a minimum bill of materials.



Each page in the PDF

Rasters generated by the RIP

Printed pages

Selection based on previous purchase

Selection based on recipient address

Selection based on recipient address and previous sales contact

One for all Letters

One for each recipient

Small set of images

Small set of dealer details

Small set of salesmen

Post-RIP raster composition

GLOBAL GRAPHICS® software

A0138

- Pre-rendered re-used elements are stored in RAM within the RIP and combined with variable data for each page at rendering time. This produces a very significant performance gain, but is very easy to implement into a new DFE because all of the work is performed within the RIP and it does not require any technology from other parties.
- Rasters for re-used and variable data elements are delivered by the RIP with masks and metadata to allow caching technology supplied by the DFE vendor to manage them, and to aggregate them into whole-page rasters for printing outside of the RIP. This produces the highest possible performance.

The ability to configure the Harlequin VariData to work entirely within the RIP, or to export rendered elements for aggregation after the RIP, makes it a very scalable solution. It can be used in a wide variety of solutions at different price points, including allowing for field-upgrades by the addition of a hardware stitching board, for instance.

Conclusion

Thus Harlequin VariData addresses the key drawbacks of VDP-specific PDLs, while ensuring that performance can still be maximized for VDP jobs created today and into the future:

a) The same PDF data stream can be submitted to a wide variety of DFEs and presses.

b) Viewers, preflight tools and other components are widely available for PDF, enabling easy construction of complete workflows.

c) The creator and print company can jointly agree on the level of graphical richness that's appropriate for a specific job. The use of some options for live transparency in PDF may cause a job to run slower or require additional horsepower in the DFE, but selecting PDF does not impose artificial constraints.

d) Just about anyone, with any software, can create a PDF file that will work well with Harlequin VariData.

The InfoTrends figures show the dominance of optimized PDF in variable data printing, and Global Graphics believes that trend will grow into the future, especially as PDF/VT is adopted.

But those same figures also show that formats such as PPML still have a place in the VDP mix. The Harlequin RIP's ability to process EPS, PDF, TIFF and JPEG within a single renderer, and with consistent color management makes it a perfect part of any solution addressing the GA subset of PPML, and, of course, it can also be used to process optimized PostScript.

Harlequin RIPs with both internal and external Harlequin VariData can be made available to qualified companies wishing to evaluate Global Graphics' solutions for inclusion in DFEs.

March 2012

GLOBAL GRAPHICS®
software
™

www.globalgraphics.com

Global Graphics Software Inc.
Somerset Court, Suite 320
281 Winter Street
Waltham, MA 02451, USA
Tel: +1-978-849-0011

Global Graphics Software Ltd
Building 2030
Cambourne Business Park
Cambourne, Cambridge
CB23 6DW UK
Tel: +44 (0)1954 283100
Fax: +44 (0)1954 283101

Global Graphics KK
704 AIOS Toranomon Bldg.
1-6-12 Nishishimbashi, Minato-ku,
Tokyo 105-0003
Japan
Tel: +81-3-6273-3740
Fax: +81-3-6273-3741

Exhibit 5
to Gauthier Declaration

HP Indigo Production Manager

Flexible, scalable Digital Front End power for high volume, complex jobs

Flexible, high performance RIP

If your business involves printing high volume, variable data, photo specialty or complex jobs, you can easily appreciate the need for a robust and powerful workflow solution.

The HP Indigo Production Manager off-press Digital Front End (DFE) provides high performance job processing for handling automated workflows, high volume variable data and complex static jobs. It's th ideal solution for printing environments with multiple HP Indigo presses, as well as hybrid shops with conventional and digital printing.

The HP Indigo Production Manager supports multipl HP Indigo presses and processes jobs efficiently. Its high performance scalable RIP technology enables up to 16 parallel RIPs, accommodating the most demanding production environments.

For complex "every page is different" image-rich applications such as photo calendars and photo books, HP Indigo Production Manager can efficientl manage a workflow of high volume photo specialty content to two presses concurrently. Printers can scal performance by adding additional RIPs, and can also assign RIP resources across multiple jobs or to single large jobs.

The HP Indigo Production Manager is also available in an entry level configuration. The tower DFE is a single RIP solution that extends the capability of the HP Indigo digital press 3050 by delivering additional colour management, JDF connectivity, web-to-print connectivity and PPML job processing.

Convenient, central management of multiple presses
Supporting the HP Indigo press 5000, HP Indigo press 3050 and HP Indigo press w3250, HP Indigo Production Manager enhances workflow efficiencies through centralised press control. It lets you route and view jobs on up to six separate HP Indigo presses. The solution also supports the use of automated workflows from third party software applications. As a result, customers can submit jobs that flow automatically to the RIP and then to the specified press. You can also define automated workflows for specified file types and in a multi-press environment you can assign workflows for specific press configurations.

Process variable data more efficiently
To assure reliable and efficient processing of high volumes of variable data, HP Indigo Production Manager supports the JLYT/SNAP, PPML and PPML-T variable data formats.

The solution helps you streamline workflows by supporting JDF open standard job ticket and bi-directional workflow communications–reporting job and press status and supporting open standard variable data output.

Accommodating the rapid growth of digital and conventional processes combined into single, unified production environments, the HP Indigo Production Manager effectively streamlines hybrid production workflows. It connects to Agfa :ApogeeX via the :ApogeeX Integration Pack for HP Indigo presses and to Heidelberg Prinect via the Prinect Digital Print Manager to HP Indigo presses.

Automate colour management using a rich colour gamut
The HP Indigo Production Manager lets you take full advantage of the expanded colour gamut capabilities of your HP Indigo presses, including:

PANTONE® support

- HP Professional PANTONE emulation (provides PANTONE emulation of coated, matte and uncoated PANTONE formula guides for any HP Indigo substrate using CMYK inks)
- PANTONE-licensed CMYK lookup table
- PANTONE-licensed HP IndiChrome lookup table

HP Professional Colour technologies and colour management features

- Intelligent, automatic and customisable RGB colour conversions
- CMYK colour management for emulation of standards and other devices
- ICC v.4 profile support

With tools such as HP Professional Colour technologies, HP IndiChrome and support for up to seven colours, HP Indigo Production Manager significantly automates colour management. For true offset quality and photo quality printing, printers can also use the 5th, 6th, and 7th colour ink stations for spot and special colours.

Architecture

	Tower unit (1 RIP)	**2-RIP unit (expandable to 8)**	**8-RIP unit (expandable to 16)**
Hardware	HP ProLiant ML370 Server • Single 3.0 GHz dual core processor • 3x 146GB SAS HDD • 2GB memory • HP DVD+RW 16 Drive	22U Rack 1x DL380 server • 2x 3.0 GHz dual core processor • 8x 72 GB SAS HDD • 4GB memory • HP DVD+R/RW 8x Slim	22U Rack 1x DL380 server • 2x 3.0 GHz dual core processor • 14x 72 GB SAS HDD • 4GB memory • HP DVD+R/RW 8x Slim
	HP 17" Flat Panel monitor	Rack-mounted keyboard and 17” monitor	Rack-mounted keyboard and 17” monitor
		1x HP BladeSystem p-Class Server Blade Enclosure and Power Bundle with 8 ProLiant Essentials Rapid deployment licenses	2x HP BladeSystem p-Class Server Blade Enclosure and Power Bundle with 8 ProLiant Essentials Rapid deployment licenses
		2x HP BL25p Blade servers • Single 2.8 GHz processor • 1GB memory • 36GB 15K SCSI HDD	8x HP BL25p Blade servers • Single 2.8 GHz processor • 1GB memory • 36GB 15K SCSI HDD
		Expandable to total of 8 Blade servers with purchase of optional RIP kits	Expandable to total of 16 Blade servers with purchase of optional RIP kits
	Localised power options: NA/Japan, INTL	Localised power options: NA/Japan, INTL	Localised power options: NA/Japan, INTL
Software	• Windows 2003 Server Std. Edition • Symantec Antivirus • HP Indigo Production Manager software • HP Production RIP software license	• Windows 2003 Server Std. Edition loaded on each server • Symantec Antivirus • HP Indigo Production Manager software • 2x HP Production RIP software license (expandable to 8 RIPs with purchase of optional RIP kits)	•Windows 2003 Server Std. Edition loaded on each server • Symantec Antivirus • HP Indigo Production Manager software • 8x HP Production RIP software license (expandable to 16 RIPs with purchase of optional RIP kits)
	Languages: • English • French • Italian • German • Spanish • Japanese • Chinese (simplified)	Languages: • English • French • Italian • German • Spanish • Japanese • Chinese (simplified)	Languages: • English • French • Italian • German • Spanish • Japanese • Chinese (simplified)
Support for multi-press	Single press only	Yes	Yes
Dimensions:			
Height	46.99 cm	111.35 cm	111.35 cm
Depth	71.75 cm	100.82 cm	100.82 cm
Width	22.86 cm	61.28 cm	61.28 cm
Weight (uncrated)	39.5 kg	225.4 kg	368.3 kg
Power requirements (international)	1x dedicated circuit	1x dedicated 30 amp circuit supplying one-phase of 208-240 VAC which supports IEC 309-32A cord cap	2x dedicated 30 amp circuit supplying one-phase of 208-240 VAC which IEC 309-32A cord cap

HP Indigo Production Manager

Network support

2- and 8-RIP units:

Internal multi-Gbit network:

- 2-RIP unit has 4 Gbit connection to RIPs; 2 Gbit connection to HP digital press(es)
- 8-RIP unit has 4 Gbit connection to RIPs; 4 Gbit connection to HP digital press(es)
- 2-RIP unit has a managed ProCurve 24 port Gbit switch
- 8-RIP unit has a managed ProCurve 48 port Gbit switch

Tower unit:

- NIC 10/100/1000 LAN

Internet & network connectivity requirements

Internet:	Broadband connectivity required; minimum 256 kbps, recommended 512 kbps or better
Network type:	100Base-T, preferably 1Gbit
Network connectivity:	CAT-5e or better copper-based network cable
IP address:	Static IP address for the server, or DHCP assigned IP address with permanent lease

Remote Production Manager requirements

Client software runs on Windows XP and Mac OS 10.x

Options

HP Indigo Production Manager RIP kit (for 2- and 8-RIP Units only); Each RIP kit includes:

- 1x HP BL25p Blade Server
- 1x Microsoft Windows 2003 Server Std. Edition
- 1x Symantec Antivirus
- 1x HP Production RIP software license (authenticated at installation)

The HP Indigo Production Manager currently supports the HP Indigo presses 5000, 3050 and w3250. Future versions will support other HP Indigo presses.
Specifications subject to change without notice.

North America
Hewlett-Packard Company
20 Perimeter Summit Blvd.
Mail Stop 401
Atlanta, GA 30319
USA
Tel: +1 800 289 5986
Fax: +1 404 648 2054

Europe, Middle East and Africa
Hewlett-Packard Company
Limburglaan 5
6221 SH Maastricht
The Netherlands
Tel: +31 43 356 5656
Fax: +31 43 356 5600

Asia Pacific
Hewlett-Packard Company
138 Depot Road
Singapore 109683
Tel: +65 6727 0777
Fax: +65 6276 3160

Latin America
Hewlett-Packard Company
5200 Blue Lagoon Drive
Suite 950
Miami, FL 33126
USA
Tel: +305 267 4220
Fax: +305 265 5550
informahpindigo@hp.com

Israel
Hewlett-Packard Company
Kiryat Weizmann
P.O. Box 150
Rehovot 76101
Israel
Tel: +972 8 938 1818
Fax: +972 8 938 1338

www.hp.com/go/graphic-arts

4AA1-0277EEE, February 2007

Exhibit 6
to Gauthier Declaration

O’Neil Data Systems

HP Indigo presses power targeted marketing campaigns

“Our Indigo presses are allowing us to produce and deliver products that couldn’t be made just a few years ago.”
— Jim Lucanish, General Manager, O’Neil Data Systems

HP customer case study: O’Neil Data Systems positions itself for growth in 1:1 marketing with move to HP Indigo presses, HP data infrastructure
Industry: Customized 1:1 marketing

Objective:

Improve digital printing quality and throughput to enable high-volume, customized document creation and output for new marketing campaigns.

Approach:

O’Neil Data Systems switched its color digital printing workflow to HP Indigo presses and all its data front-end to HP BladeSystems and storage.

IT improvements:

- Improved RIP processing.
- Faster access to stored customer data.

Business benefits:

- Faster print throughput.
- Improved color consistency and quality.
- Scalability to grow with client needs.

For more than 30 years, O’Neil Data Systems has been delivering technology-driven marketing communications services for its clients. It pioneered targeted, 1:1 marketing before there were data/image merge capabilities or digital printing of any kind, and has been an HP computer customer. So when HP acquired Indigo press technology several years ago and began expanding its capabilities, the company knew great things lay ahead.

“What HP has done with Indigo technology is really amazing,” says Jim Lucanish, General Manager for O’Neil. “Our HP Indigo presses are helping us do things for clients that simply couldn’t be done a few years ago. And with quality that could only be produced on litho presses until now.”

New solution for a proven leader

O'Neil is a leading provider of marketing communication and publishing solutions. Its customer-centric applications and services include electronic document delivery, web applications, high-speed digital printing in both color and black and white, automated composition, offset printing, warehousing and fulfillment services.

> *"What HP has done with Indigo technology is really amazing. Our HP Indigo presses are helping us do things for clients that simply couldn't be done a few years ago. And with quality that could only be produced on litho presses until now."*
> Jim Lucanish, General Manager, O'Neil Data Systems

Every day, O'Neil produces and delivers millions of time-sensitive documents both in electronic and paper media. Its clients include Visa, Federal Express, Humana, Blue Cross, Federal Express, Toyota, EarthLink and Kaiser Permanente.

A generation ago, the company produced stock market reports literally by hand. The report on each company was produced with a combination of 35mm film imaging and high-contrast litho film and plates, then hand-gathered for each investment client.

"Then Bill O'Neil said we needed to produce 12,000 pages a day. We literally brought in a rocket scientist who found a way to employ a microfilm plotter," recalls Operations Manager Steve Ellithorpe. "This was before Postscript, or any other page description language, even existed. It was all mad scientist stuff."

Over the years, the company adopted every technology advance it could, always pushing the limits. Now fast forward 30 years to the present. O'Neil is still producing customized, data-intensive documents for print. But it's doing so faster, in much greater quantities, and with dramatically improved quality.

HP technology enables new solutions

"Our two HP Indigo presses produced 10 million impressions of custom documents in the past three weeks with incredible color quality," says Lucanish. "Company-wide, including black-and-white impressions, we did about 85 million letter-sized impressions."

One of the company's biggest markets is personalized healthcare documents. When Medicare Part D went into effect, O'Neil provided mailings to a third of the senior Medicare recipients. "Major healthcare providers need to get a report out to thousands of members summarizing their individual accounts, plans and benefits," explains Lucanish. "Using our HP Indigo presses, we did 4.5 million letter-sized impressions for one healthcare company's pre-enrollment push, all as 64-page, full-color, glossy reports personalized to the individual."

> *"At one point, it was taking us eight hours to RIP a file. We went from that to about 10 minutes. HP technology can be truly amazing."*
> Operations Manager Steve Ellithorpe

The workhorse for such projects is an HP Indigo press w3250, a 6/6 web press that O'Neil mated to a custom inline finishing system. The company also employs an HP Indigo 5000 sheet-fed press for covers, jobs requiring changes in paper stock, and shorter runs. The company's sophisticated data system for storing, accessing and transforming data is driven by

Customer solution at a glance

Primary application
Customized 1:1 marketing document creation

Primary hardware
- HP Indigo press 5000
- HP Indigo press w3250
- HP Indigo Production Manager
- HP Designjet Z2100 Photo Printer
- HP BladeSystem with ProLiant BL680c server blades
- HP StorageWorks EVA6000 storage array

“We find the quality we’re getting from our Indigo presses is far superior to other digital, or short-run printing.”

Jim Lucanish, General Manager,
O’Neil Data Systems

HP ProLiant blade servers and an HP StorageWorks EVA6000 storage array. The variable data stream and images are fed into an HP Indigo Production Manager off-press digital front end with 8 RIPs to ensure the press is never kept waiting.

As a result, O’Neil is able to produce truly customized healthcare documentation that is still cost-effective. In the past, a health insurer would ask O’Neil to produce a non-personalized, thick Welcome Kit for their members. These large, multi-page documents were costly to produce and very expensive to mail. “By the time you put that all together, you were mailing a two-pound digital book,” says Ellithorpe. “What we’re doing now is cutting down their page count. We saved one of our digital clients $600,000 in postage alone. So customized digital printing is driving down their overall cost.”

“What HP has brought to the table is expertise in a combination of technologies, from printing to servers, storage and networking, that gives us a complete solution.”
Operations Manager Steve Ellithorpe

HP BladeSystems even power the RIPs for O’Neil’s non-HP black-and-white digital printers. “At one point, it was taking us eight hours to RIP a file,” recalls Ellithorpe. “We went from that to about 10 minutes. HP technology can be truly amazing.”

In fact, he says that while Indigo presses have provided a major step forward in digital print quality and throughput, the company’s investment in the data front-end — powered by HP servers and storage — is equally important in allowing O’Neil to stand out from the competition. “What HP has brought to the table is expertise in a combination of technologies, from printing to servers, storage and networking, that gives us a complete solution.”

“HP Indigo presses have unbelievable color and incredible uptime, so we’re able to deliver fantastic quality even under tight deadline. That’s why clients are asking for more and more color all the time.”
Jim Lucanish, General Manager, O’Neil Data Systems

HP Indigo quality pays

Historically, custom, one-up printing was not just expensive, but inconsistent. HP Indigo presses put an end to that problem. “We are now able to provide our clients a viable solution to create personalized, digital color documents with consistent offset quality,” says Mark Rosson, Vice President, Sales for O’Neil Data Systems. “In the past, marketing departments and ad agencies would never produce high-image materials using digital color. Quality issues and consistency remained stumbling blocks. Our HP Indigo presses have changed their mindset, opening a whole new world of possibilities using personalization and variable content to otherwise static documents.”

In some ways, Lucanish adds, it’s even better than traditional lithography. “If a litho press was being run perfectly, you could get 1 percent dot gain. But with the Indigo we get 1 percent every time. And we learned very quickly that the quality is well worth the purchase price.”

When the company produced a furniture catalog on another brand of digital printer, inconsistency in the color quality rendered the job almost unusable. "We had done a color proof on the same printer, which the customer approved. But when we printed the catalogs, the hue of this green couch varied all over the place. They were very unhappy," says Lucanish.

"With HP Indigo presses, you don't have to think about quality. The quality is good — every impression, through the length of the job. In one of our tours with the health insurance client, we showed them output from the Indigo and that led to them doing a job of 10 million impressions with us."

To complement the company's two HP Indigo production presses, it has also deployed an HP Designjet Z2100 Photo Printer, which provides 44-inch-wide, large format proofing capability that's calibrated to the production presses.

The bottom line: when customers ask, 'Can you do this?' the staff at O'Neil routinely says yes. "It's often something they wanted to do two, three or four years ago but couldn't," says Lucanish. "Our Indigo presses are allowing us to produce and deliver products that couldn't be made just a few years ago."

Looking ahead

Lucanish says the company's next color digital press will be the HP Indigo press 5500. "The 5000 already gives us excellent quality for sheet-fed work, but the 5500 takes quality even further," he says. The

availability of light cyan and light magenta inks improves photo reproduction, while the addition of On-press Fast Ink Replacement should improve throughput.

"Back when HP bought Indigo we became very interested in the future of this technology," he says. "Now the improvements just keep coming. HP Indigo presses have unbelievable color and incredible uptime, so we're able to deliver fantastic quality even under tight deadline. That's why clients are asking for more and more color all the time."

To learn more, visit www.hp.com

4AA1-7263ENW, December 2007

invent

Exhibit 1
to Raasch Declaration

HP Indigo Yours Truly Designer version 7

User guide

HP Indigo Yours Truly Designer 7

User guide

The information contained herein is subject to change without notice.

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Part number:
CA294-00814
First Edition:
February 2007

THIS SOFTWARE PACKAGE HAS NOT BEEN WRITTEN, REVIEWED, OR TESTED BY QUARK, THE QUARK AFFILIATED COMPANIES, OR THEIR LICENSORS. QUARK, THE QUARK AFFILIATED COMPANIES, AND THEIR LICENSORS MAKE NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED SOFTWARE PACKAGE, ITS MERCHANTABILITY, OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. QUARK, THE QUARK AFFILIATED COMPANIES, AND THEIR LICENSORS DISCLAIM ALL WARRANTIES AND CONDITIONS RELATING TO THE SOFTWARE PACKAGE WHETHER EXPRESS, IMPLIED, OR COLLATERAL, INCLUDING, WITHOUT LIMITATION, ANY WARRANTY OF NON-INFRINGEMENT, COMPATIBILITY, OR THAT THE SOFTWARE IS ERROR-FREE OR THAT ERRORS CAN OR WILL BE CORRECTED.

IN NO EVENT SHALL QUARK, THE QUARK AFFILIATED COMPANIES, OR THEIR LICENSORS BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, CONSEQUENTIAL, OR PUNITIVE DAMAGES, INCLUDING, BUT NOT LIMITED TO, ANY LOST PROFITS, LOST TIME, LOST SAVINGS, LOST DATA, LOST FEES, OR EXPENSES OF ANY KIND ARISING FROM INSTALLATION OR USE OF THE SOFTWARE OR ACCOMPANYING DOCUMENTATION IN ANY MANNER, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY. IF, NOTWITHSTANDING THE FOREGOING, QUARK, THE QUARK AFFILIATED COMPANIES AND/OR THEIR LICENSORS ARE FOUND TO HAVE LIABILITY RELATING TO THIS SOFTWARE PACKAGE, SUCH LIABILITY SHALL BE LIMITED TO THE FEES PAID BY END USER TO QUARK, IF ANY, WITHIN THE THREE-YEAR PERIOD PRECEDING THE CLAIM, FOR THE LICENSE OF THE SPECIFIC QUARK PRODUCTS (EXCLUDING ANY THIRDPARTY COMPONENTS ADDED BY END USER OR ANY THIRD PARTY, INCLUDING DEVELOPER OR AN INTEGRATOR), OR THE LOWEST AMOUNT UNDER APPLICABLE LAW, WHICHEVER IS LESS. THESE LIMITATIONS WILL APPLY EVEN IF QUARK, THE QUARK AFFILIATED COMPANIES, AND/OR THEIR LICENSORS HAVE BEEN ADVISED OF SUCH POSSIBLE DAMAGES.

Contents

1 Introduction

2 Installing or upgrading YTD

3 Preparing YTD jobs

4 Working with YTD rules

5 Imposing YTD jobs

6 Creating the output file

7 Changing YTD preferences

8 SNAP fonts

9 DUPLO Finisher support

1 Introduction

This chapter contains the following topics:

- Overview
- System requirements
- Basic concepts
- YTD palettes

Overview

Welcome to the HP Indigo Your Truly Designer User Guide.

HP Indigo Yours Truly Designer is your gateway to personalization job creation. It is a flexible tool that enables fine-tuning of your workflow to suit the job's nature and your workflow routines. You can also use it as an imposition tool for all your jobs, whether they include personalization or not.

Yours Truly Designer (YTD) is a QuarkXPress XTensions Software for the Macintosh. With it you can create, integrate, and preview all personalization jobs, whether they contain fixed data or variable text/image elements (personalization channels). You can create intricate imposition matrices to fit any job type: simple or personalization, step & repeat, or imposed books.

The HP Indigo Yours Truly Designer User Guide is divided into nine chapters:

1 "Introduction" describes the purpose of YTD and explains YTD basic concepts such as personalization, job layouts, file descriptions, and job workflows.

2 "Installing or upgrading YTD software" explains how to install or upgrade YTD.

3 "Preparing YTD jobs" describes how to create YTD jobs.

4 "Working with YTD rules" explains how to use rules to add conditions to text and image personalization channels.

5 "Imposing YTD jobs" describes how to impose the YTD job, including spread elements, how to create imposition templates, and how to send an imposition template to the press.

6 "Creating the output file" explains how to create output files as JLYT or PPML, to be sent to press for printing, or a PDF file for preview and proofing.

7 "Changing YTD preferences" explains how to setup the default values for elements used by YTD.

8 "SNAP fonts" describes how to create SNAP fonts.

9 “Duplo finisher support” summarizes the way to include finishing parameter barcodes in your job for use with Duplo finishers.

10 "Service and support" provides contact details for obtaining service and support.

Conventions used in this guide

This guide uses the following documentation conventions:

- Keyboard keys appear in all capital letters. For example: Press the SHIFT key.
- Window names appear in italics. For example: Enter the following information in the *Modify* window.
- Menu names and menu options are indicated in **bold** type. For example: Select **New** from the **File** menu.
- Buttons and options to click appear in **bold** type, for example: After selecting the options in the *Print* window, click **OK**.
- References to other sections in the guide appear in quotes, for example: See the “Installation” section on page 24.
- Screen messages and text that you are expected to type are displayed in a different font. For example: `When the Installation completed message appears...`

Window panes

This manual includes descriptions of windows. A window may contain several panes. Each pane uses the naming conventions indicated in figure 1-1.

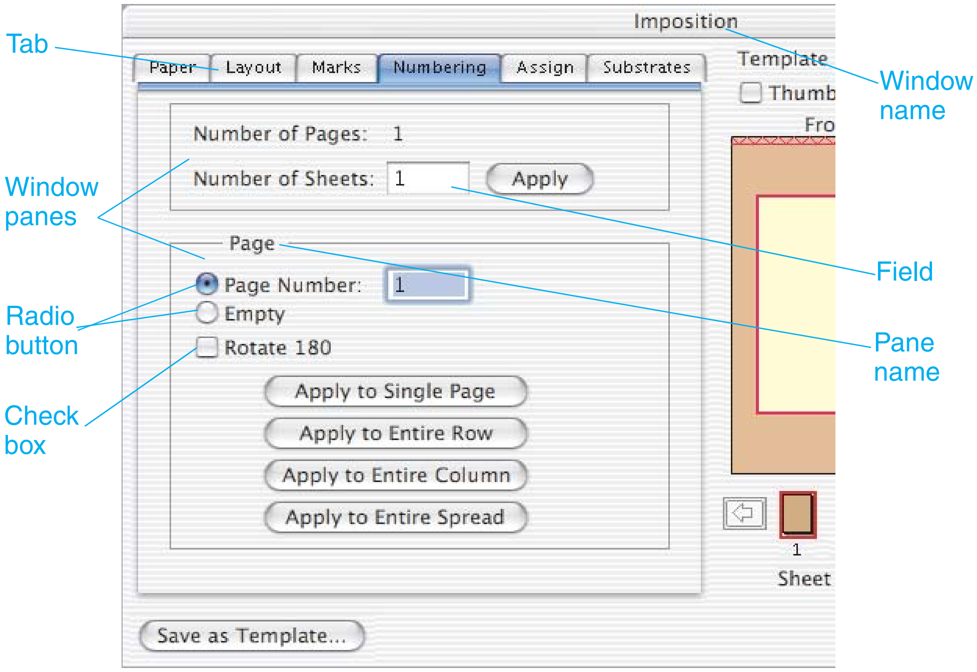


Figure 1-1. Example window

System requirements

- Macintosh Operating System version 10.3 or greater
- QuarkXPress version 7
- HP Indigo press with software version 5.0 or later for printing the YTD output file JLYT format.
- HP Indigo press 5000 or HP Indigo production flow for printing the YTD output file JLYT and PPML format.

Basic concepts

Personalization

Personalization refers to printing variable data together with fixed data, giving the effect of personalized printed sheets. A common form of personalization is a word processor mail-merge application. In this case, a standard letter is prepared and variable changing data, such as names and addresses, are inserted in the standard letter. The result is a batch of letters, personalized to the various addressees.

YTD's personalization expands on this simple mail-merge concept by combining variable color images, as well as text, with fixed images and text, using sophisticated page composition.

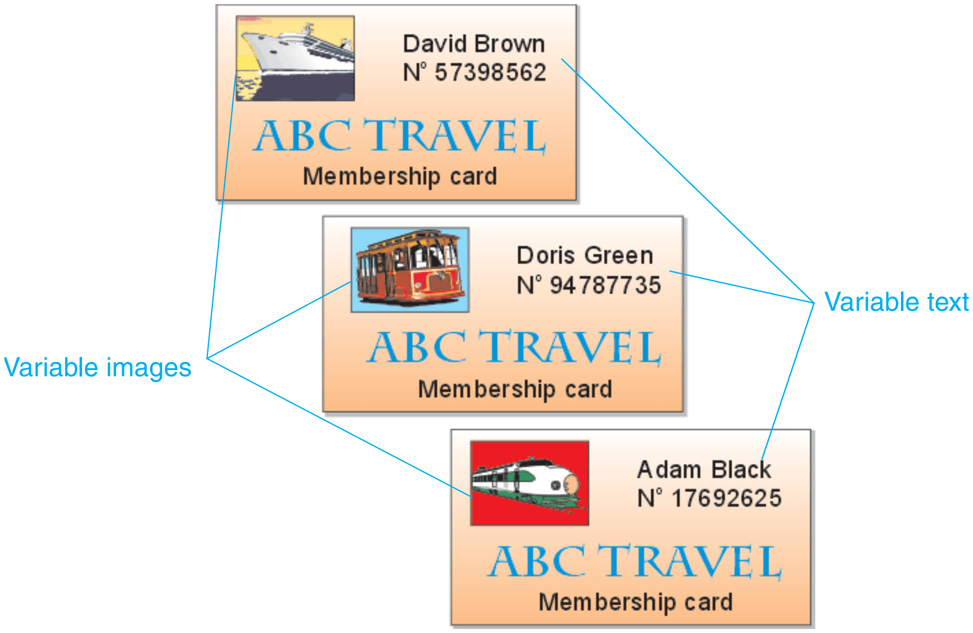


Figure 1-2. Personalization

Personalization channels

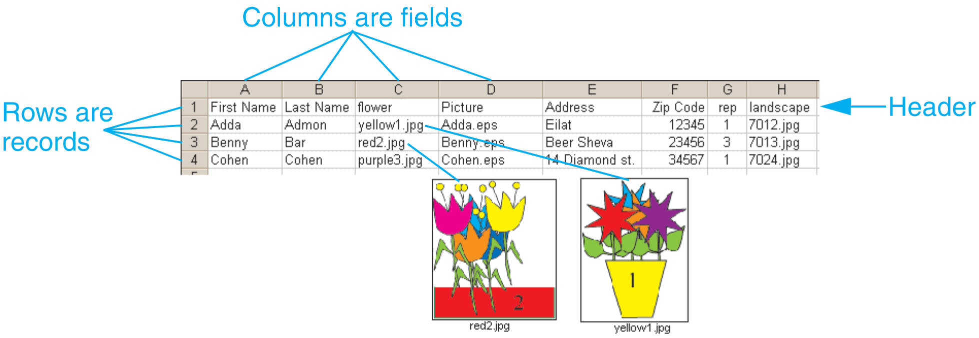
The personalization channels contain image or textual data that can change for each copy of the page. The channel in each such copy represents a channel cycle. A channel assigns data through a database field, or inputting fixed text into QuarkXPress, or through a multi-page document. Data may include images, fixed text, and variable text. Text channels may contain data for both fixed text and variable text.

Note

If your HP Indigo press is authorized for monochrome personalization only, you must prepare the image and text channels in monochrome. In any given monochrome personalization job, all channels must be of the same separation.

Databases

The linkage between a personalization channel and variable data is through the assignment of a database field to the channel. The database field is part of a database report file that is a text file arranged in tabular form. The table rows are *records* and the columns are fields. The table header, usually the first record in the file that contains the field (column) names, is the *database* header.



	A	B	C	D	E	F	G	H
1	First Name	Last Name	flower	Picture	Address	Zip Code	rep	landscape
2	Adda	Admon	yellow1.jpg	Adda.eps	Eilat	12345	1	7012.jpg
3	Benny	Bar	red2.jpg	Benny.eps	Beer Sheva	23456	3	7013.jpg
4	Cohen	Cohen	purple3.jpg	Cohen.eps	14 Diamond st.	34567	1	7024.jpg

Figure 1-3. Database report file and associated image files

Since a channel can contain image or textual data, the database field from which the channel gets its data can be actual textual data or filenames of image data (or both, in special cases). The database column, represented by the database field, must refer to the same data type (either all text values or all image filenames). In the above example, columns A and B contain variable text; all the fields in the columns are straight text. Columns C and H contain variable image data; all the fields in these columns are filenames of JPG image files.

The order of the records in the database determines the print sequence: record one is the first to be printed in the channel, record two is second, and so on. Each record of the database file represents one personalization copy.

The DB report file must be field delimited, either as a simple text file (txt) or as a field delimited Excel file (txt or csv). Common delimiters are comma (,), TAB, space, and semicolon (;).

YTD workflows

Imposition only

This mode provides the ability for jobs with no personalization channels to utilize the advanced features of YTD imposition. A job using this mode contains only QuarkXPress pages imposed on the spreads.

Personalization

This mode is for jobs that have personalization channels such as text channels, image channels, or document channels. The text and image channels can be derived from database files. Document channels can be derived from multi-page QuarkXPress or PostScript documents.

The wide range of QuarkXPress typography options (such as text paths and attributes) and image options (all picture box shapes and QuarkXPress-recognized image file types) are available.

SNAP (**S**wift **N**ative **A**ccelerated **P**ersonalization) is the technology embedded in HP Indigo presses that allows for the assembly of text or image data at full print speed, without needing to RIP it. This technology accelerates the processing of variable information jobs since SNAP image personalization channel data and SNAP text personalization channel data is not processed to PostScript.

With HP Indigo SNAP, the press accepts the information from a database and converts it into text without any pre-processing. Pictures in JPG and TIFF formats only are processed automatically, producing printable images without requiring any other intervention.

Personalization text or image channels can carry any QuarkXPress attribute. The channels are processed as SNAP channels, or converted to PostScript depending on their attributes. SNAP channels can contain only specific attributes.

YTD personalization jobs can be set as:

- SNAP jobs which contain only SNAP channels.
- Non-SNAP jobs which contain only non-SNAP channels.
- Mixed channel jobs which contain both SNAP and non-SNAP personalization channels.

SNAP jobs

SNAP jobs are jobs that contain only SNAP channels.

SNAP job output creation is very fast since personalization channels that are suitable for SNAP jobs do not need to be processed into PostScript files to be embedded into the final output file.

Consequently, the personalized data in the SNAP job is not RIP'd at the press. The variable images are converted directly into printable format. Variable text data is not included in the output file but rather it is added on-the-fly while printing at the press.

The SNAP jobs output file can be either a Template type, or a Job type.

Template type outputs does not include any embedded database file or variable data. They are useful for multi-use jobs where the same job may need to be printed more than once, each time using a different DB report file.

Job type outputs contain an embedded database, but not embedded variable data. They can be printed with that database only.

Non-SNAP jobs

Non-SNAP jobs are jobs whose output file includes an embedded database report file and embedded variable data. Non-SNAP jobs can be printed with the embedded DB report data only.

A non-SNAP job may contain a mixture of channels that are acceptable for SNAP jobs and other channels that are not acceptable for SNAP jobs. The channels are processed differently. The non-SNAP channels are processed into PostScript files that are embedded into the output file, while the template channels do not generate PostScript files.

Since the output file contains embedded PS files that increase its size, the file creation and transfer times take longer than for SNAP jobs. Additionally, RIP is performed at the press for all the embedded PS files.

Non-SNAP output files can be only of the Job type.

Rules

Rules provide an added dimension in the personalization workflow. The appearance and content of fixed text, variable text fields, and variable image channels can be selectively modified using rules.

Image Rules allow the printing of different images based on the value or contents of a database field. Text Rules allow the change of appearance and content for fixed text or a text field (font, color, size, or content).

YTD supplies a set of predefined text, field, and Image Rules. These rules can be modified as required, or new rules can be created to match your personalization job needs.

YTD job components

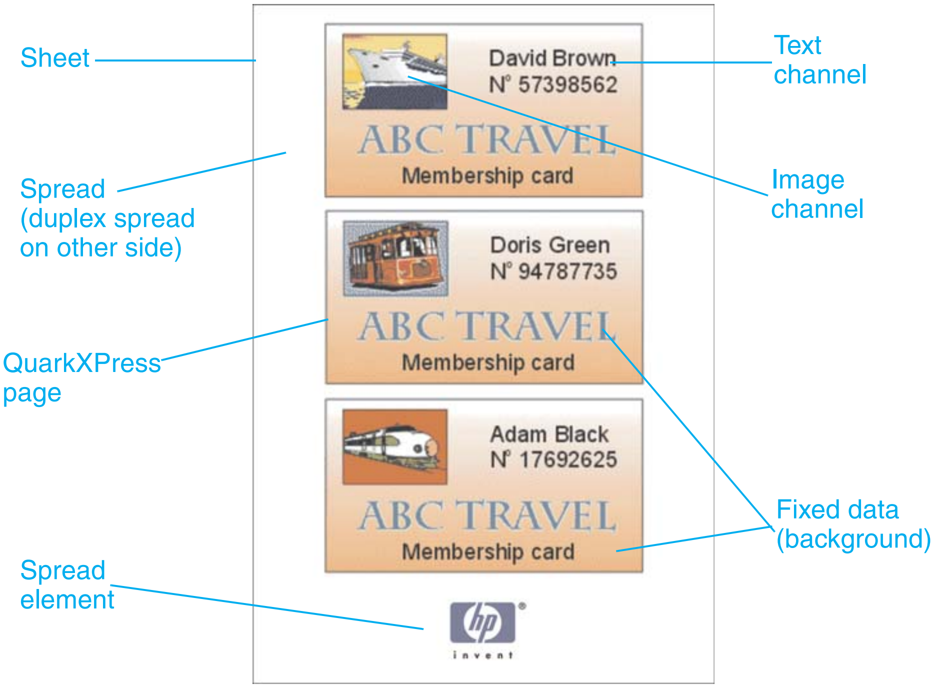


Figure 1-4. Personalization job contents

YTD layers

A YTD job arranges data into two major layers:

1. Fixed data and Personalization channels arranged in any layer order, depending on the design requirements.
2. Spread elements (graphics, text, images) and crop marks added to the spread after imposing QuarkXPress pages as a separated layer.

Note Rounaround is applicable between channels and fixed data elements. It is not applicable between the two major layers.

Note Using the job editor on the press, you can change the positioning of the layers.

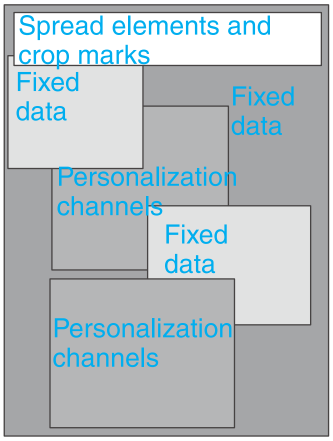


Figure 1-5. YTD layers

RUT file

When YTD creates an output file, it automatically generates or updates the job's reuse table (RUT) file. The RUT file contains information on the job's reusable elements.

Reusable elements

A reusable element (text, image, background, and so on) recycles for jobs sent to press more than once, or elements used multiple times in the same job, without the need to recreate the element in YTD, and without the need to RIP it or load it again at the press.

YTD recognizes all fixed data as one reusable element. Additionally, image channels (when defined as reusable) are reusable elements.

Output formats

JLYT output

JLYT format is the HP press proprietary input format. It enables the full use of HP Indigo Press features and optimizations, as the SNAP technology embedded in the presses. Outputting JLYT will maximize the use of YTD optimizations for the HP Indigo Presses.

PPML output

PPML is an industry format that enables scalable RIP capability, which allows parallel processing, for faster processing of personalization jobs. The PPML output is recommend if RIP includes the scalable RIP option, and the job requires heavy processing; or when the PPML imposition options are used in the YTD job definitions. HP Indigo press 5000 and HP Production Flow supports PPML format and is capable of scalable RIP configuration.

PPML output created by YTD complies with PPML Templating Specification, Version 1.0 (PPMLT), defined by the Print on Demand Initiative (PODi), a PPML working group. PPMLT, as with JLYT format, creates template jobs with the ability to reuse the template using a different database each time. Unlike PPML, PPMLT enables the definition of long runs of personalized content without creating and sending large amounts of repetitive data.

PDF output

PDF is a common format for viewing output. Exporting a YTD job as a PDF is recommended for sending a few records for approval by a client or to proof the output on a low resolution printer.

Special features with HP Indigo press 5000

YTD 7 includes special features supported only by HP Indigo press 5000 and HP production flow:

- Multi-substrate definition (HP Indigo press 5000 only)
- PPML imposition
- PPML output file

All other YTD capabilities operate with all HP Indigo presses.

Using HP IndiChrome and PANTONE® colors in YTD

The HP IndiChrome color model uses six printing inks: the conventional CMYK, plus two special inks—orange and violet.

RGB images

- Fix background RGB EPS images may be converted to HP IndiChrome colors.
- RGB variable images in TIFF, JPG, and EPS format may be converted to HP IndiChrome colors.
- The conversion of images from RGB to HP IndiChrome colors is done outside YTD by the RIP and press software.

PANTONE colors

PANTONE colors can be added to the QuarkXPress Colors palette, as follows:

1. Click **Edit** and **Colors**.
2. In the QuarkXPress *Colors* window that appears, click **New**.
3. In the **Model** drop-down menu, select **PANTONE Coated**.
4. Select the relevant PANTONE color.
5. Click **OK**.
6. In the *Colors* window, click **Save**.

When using the HP IndiChrome option, note the following:

- An EPS file that contains PANTONE colors and is created in FreeHand or Illustrator can be imported into a QuarkXPress document. The PANTONE colors that are used in the EPS file are converted by YTD to HP IndiChrome colors.
- Text and graphic elements can include PANTONE colors that YTD converts to HP IndiChrome colors.

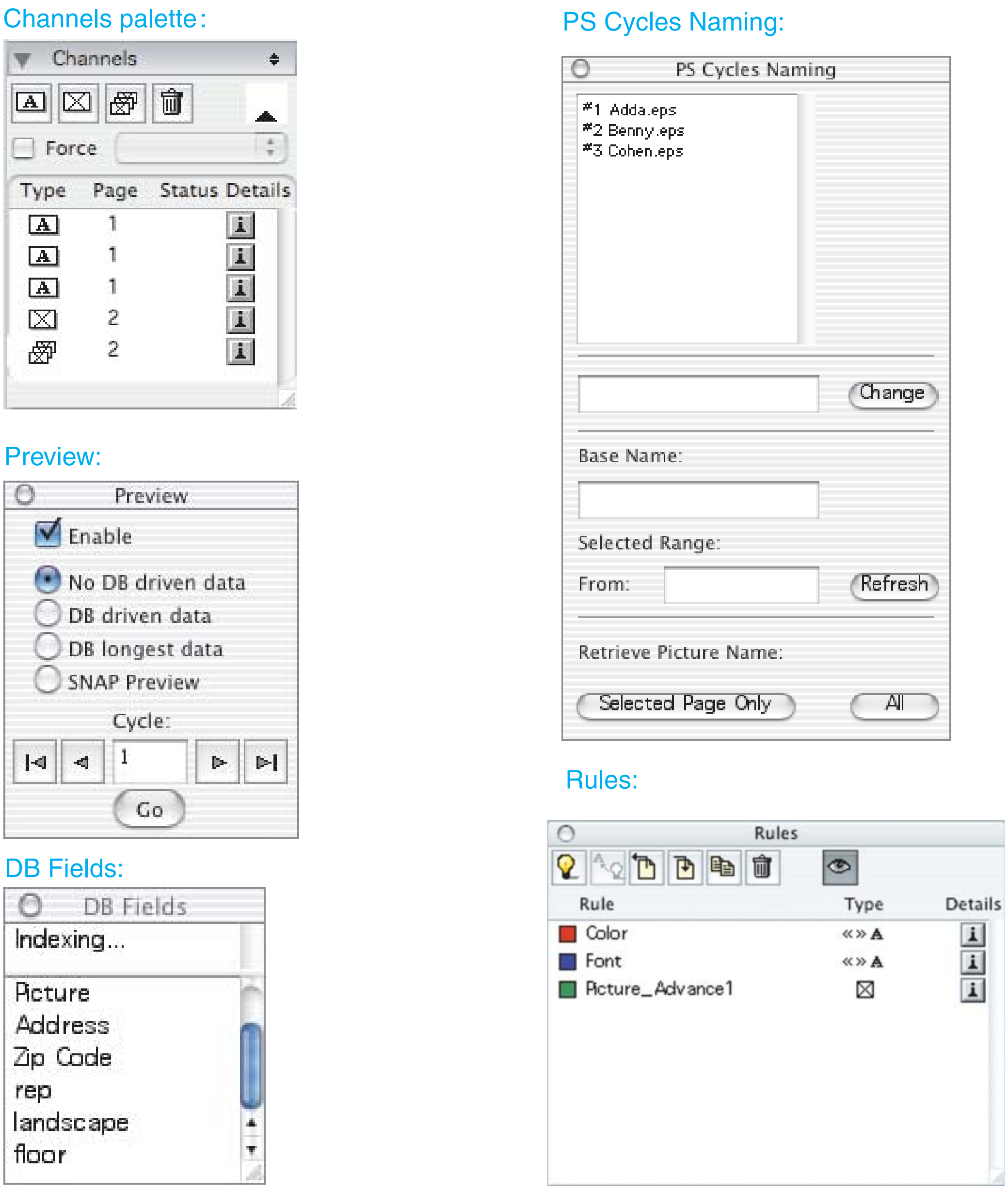
- The conversion processes for EPS files and for text and graphics elements described above are PANTONE-approved for over 1000 PANTONE colors. See the *HP IndiChrome PANTONE® simulation user guide* for more information.

Note The output file of a document that contains RGB variable images and/or graphic elements and text colored in PANTONE colors must be created in composite mode.

Note Detailed information about the HP IndiChrome system is available in chapter 8 of the *HP Indigo RIP user guide* and in the *HP IndiChrome PANTONE® simulation user guide*.

YTD palettes

YTD palettes help you define and preview channels:



Channels palette:

Type	Page	Status	Details
A	1		i
A	1		i
A	1		i
☒	2		i
	2		i

PS Cycles Naming:

#1 Adda.eps
#2 Benny.eps
#3 Cohen.eps

Preview:

☑ Enable

● No DB driven data
○ DB driven data
○ DB longest data
○ SNAP Preview

Cycle: 1

Rules:

Rule	Type	Details
🟥 Color	«» A	i
🟦 Font	«» A	i
🟩 Picture_Advance1	☒	i

DB Fields:

Indexing...

Picture
Address
Zip Code
rep
landscape
floor

Figure 1-6. YTD palettes

You can open or hide individual palettes by clicking **Yours Truly**, **Palettes**, and the individual palette name.

You can open or hide the Channels, Preview, and DB Fields palettes by clicking **Yours Truly**, **Palettes**, and **Main Palettes**.

The various YTD palettes are described in the sections of this guide in which they are used.

2 Installing or upgrading YTD

This chapter contains the following topics:

- Installing/upgrading YTD software
- Printing jobs created by previous versions

Installing/upgrading YTD software

Note You must install QuarkXPress 7 prior to installing the YTD 7 software.

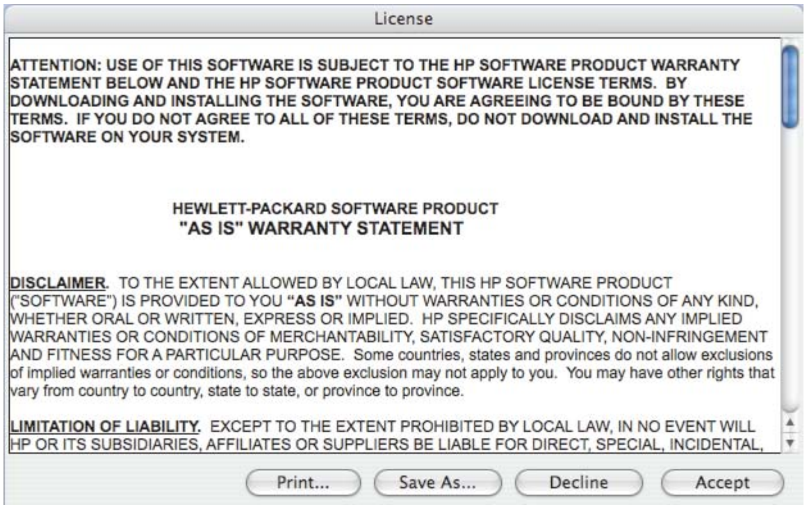
1 Quit QuarkXPress 7 application.

2 Insert the YTD software CD-ROM into the drive.

3 Double-click the CD-ROM icon to display its contents.

4 Double-click the **HP Indigo YTD 7 Installer** icon.

5 "Gathering information" appears and then the *License* window pops up.



License

ATTENTION: USE OF THIS SOFTWARE IS SUBJECT TO THE HP SOFTWARE PRODUCT WARRANTY STATEMENT BELOW AND THE HP SOFTWARE PRODUCT SOFTWARE LICENSE TERMS. BY DOWNLOADING AND INSTALLING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THESE TERMS. IF YOU DO NOT AGREE TO ALL OF THESE TERMS, DO NOT DOWNLOAD AND INSTALL THE SOFTWARE ON YOUR SYSTEM.

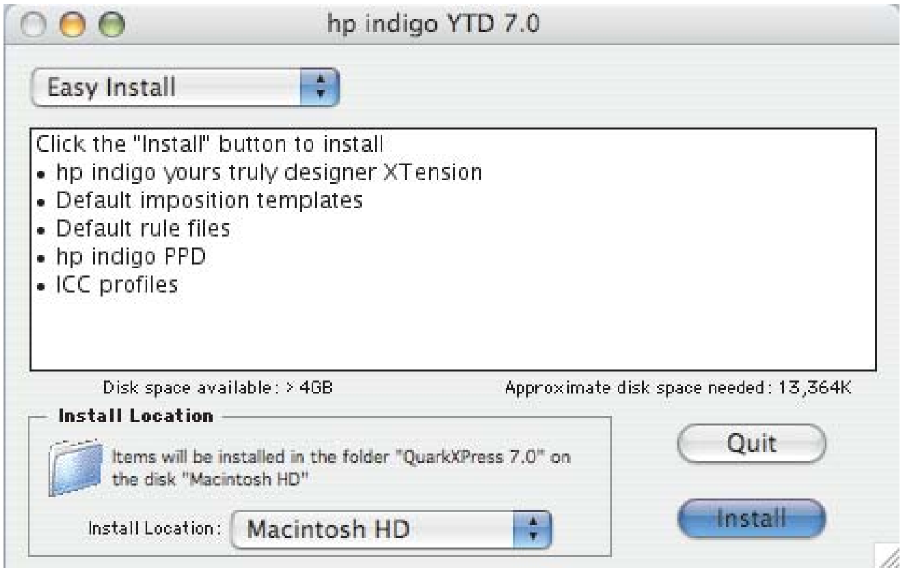
HEWLETT-PACKARD SOFTWARE PRODUCT
"AS IS" WARRANTY STATEMENT

DISCLAIMER. TO THE EXTENT ALLOWED BY LOCAL LAW, THIS HP SOFTWARE PRODUCT ("SOFTWARE") IS PROVIDED TO YOU **"AS IS"** WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED. HP SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. Some countries, states and provinces do not allow exclusions of implied warranties or conditions, so the above exclusion may not apply to you. You may have other rights that vary from country to country, state to state, or province to province.

LIMITATION OF LIABILITY. EXCEPT TO THE EXTENT PROHIBITED BY LOCAL LAW, IN NO EVENT WILL HP OR ITS SUBSIDIARIES, AFFILIATES OR SUPPLIERS BE LIABLE FOR DIRECT, SPECIAL, INCIDENTAL,

Figure 2-1. License window

6 Click **Accept**. The **HP Indigo YTD 7 Installer** window opens.



hp indigo YTD 7.0

Easy Install

Click the "Install" button to install
- hp indigo yours truly designer XTension
- Default imposition templates
- Default rule files
- hp indigo PPD
- ICC profiles

Disk space available : > 4GB   Approximate disk space needed : 13,364K

Install Location

Items will be installed in the folder "QuarkXPress 7.0" on the disk "Macintosh HD"

Install Location : Macintosh HD

Figure 2-2. Easy Install

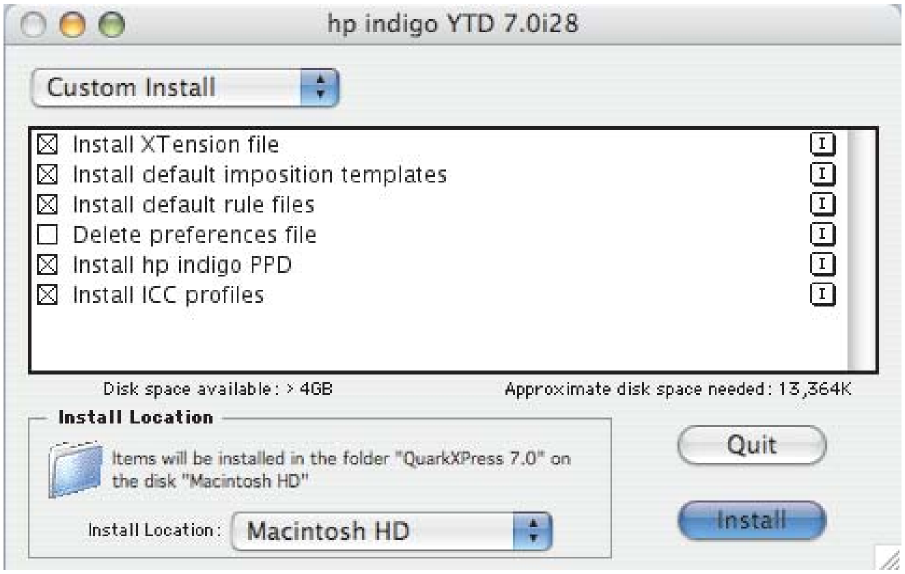
7 In the *HP Indigo YTD 7 Installer* window that appears, the installation defaults to **Easy Install**.

8 Select the install location by clicking on the **Install Location** drop-down menu, and selecting the desired folder.

9 When the options suit your installation/upgrade needs, click **Install**.

If you want to selectively install YTD components, do the following:

1 Click **Easy Install** and select **Custom Install** from the drop-down menu.



☒ Install XTension file
☒ Install default imposition templates
☒ Install default rule files
☐ Delete preferences file
☒ Install hp indigo PPD
☒ Install ICC profiles

Disk space available : > 4GB

Approximate disk space needed : 13,364K

Install Location

Items will be installed in the folder "QuarkXPress 7.0" on the disk "Macintosh HD"

Install Location : Macintosh HD

Figure 2-3. Custom Install

2 Select or clear the relevant check boxes.

3 Select the install location by clicking on the **Install Location** drop-down menu, and selecting a folder.

4 Click **Install**.

Printing jobs created by previous versions

YTD version 5.0.1, 5.5, 6.0, 6.1, 6.5.1 and 6.5 jobs

YTD version 7 supports jobs created using earlier YTD versions. To print jobs created using these versions, open the QuarkXPress document with YTD version 7. The software automatically updates the job to version 7. No additional actions are needed.

Note Save jobs created using YTD version 1.1 to YTD version 5.0.1 before opening them using YTD 7.

3 Preparing YTD jobs

This chapter contains the following topics:

- YTD workflows
- Selecting a database
- Defining personalization channels
- Previewing the variable job
- Cancelling a channel definition

YTD workflows

Use YTD to prepare personalization or imposition-only jobs.

For imposition-only jobs, follow these procedures:

- Designing your QuarkXPress document
- "Imposing YTD jobs" on page 85
- "Adding spread elements", if necessary, on page 118
- "Creating the output file" on page 119

For personalization jobs, follow these procedures:

- Designing your QuarkXPress document
- "Selecting a database" on page 20
- "Defining personalization channels" on page 24
- If necessary, "Working with YTD rules" on page 53
- "Imposing YTD jobs" on page 85
- "Adding spread elements", if necessary, on page 118
- "Creating the output file" on page 119

The following page includes a diagram of the job creation work flow.

Note After you define any personalization channel, you may define additional channels, define rules to apply to the channel, or define imposition (as described in the following chapters).

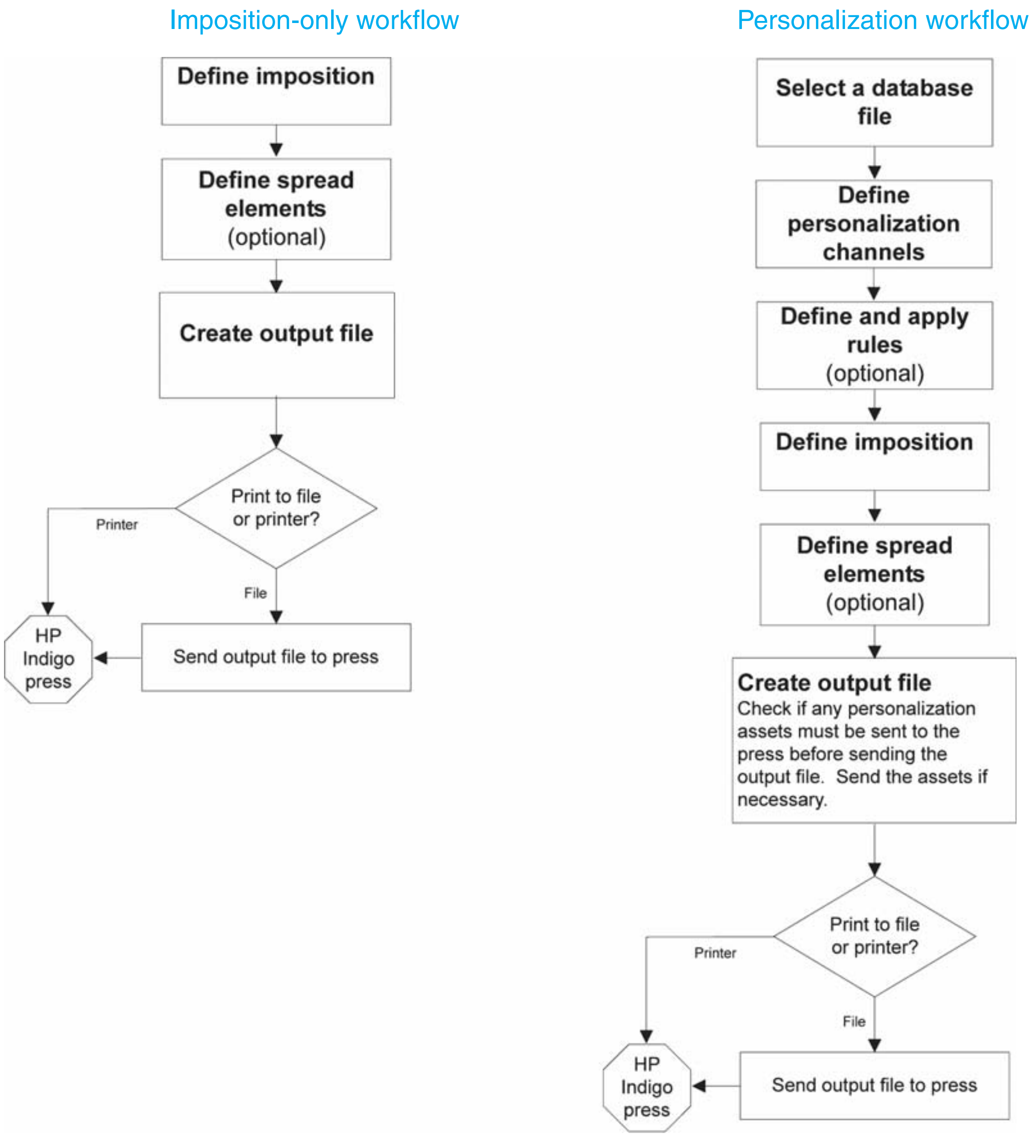

Imposition-only workflow
Define imposition
Define spread elements (optional)
Create output file
Print to file or printer?
Printer
File
HP Indigo press
Send output file to press
Personalization workflow
Select a database file
Define personalization channels
Define and apply rules (optional)
Define imposition
Define spread elements (optional)
Create output file
Check if any personalization assets must be sent to the press before sending the output file. Send the assets if necessary.
Print to file or printer?
Printer
File
HP Indigo press
Send output file to press

Figure 3-1. YTD workflows

Note

Naming conventions for files on the Macintosh and PC/Windows are different. Since YTD jobs are sent to the press' computer, which is a PC/Windows system, all file names in YTD (including job names and document names) must conform to the PC/Windows naming conventions. This means that file names can be composed of all alphanumeric and special characters except \ / * : ? | < >.

QuarkXPress Project names can have a maximum of 29 characters.

QuarkXPress Layout names can have a maximum of 26 characters.

Selecting a database

This section describes how to select and define a database file for use with a personalization job. A personalization job can contain fixed text, variable text, and variable images. In the personalization workflow, the order of printing variable text data and variable images is taken from a database file (also known as DB report file).

The DB report file must be field delimited, either as a simple text file (txt) or as a field delimited Excel file (txt or csv). Common delimiters are comma (,), TAB, space, and semicolon (;). Delimiters cannot consist of any of the following: \ / * : ? | < >.

The first line in any database file is considered to be a header line if it provides header information or actual data. Select the DB header and DB report files. You may use the same file for both, or a different file for each, as long as the column structure of the different files is identical. Both files must contain the same number of columns, in the same order; for example, if the second column in the DB header file contains the name field, the second column in the DB report file should also contain the name field.

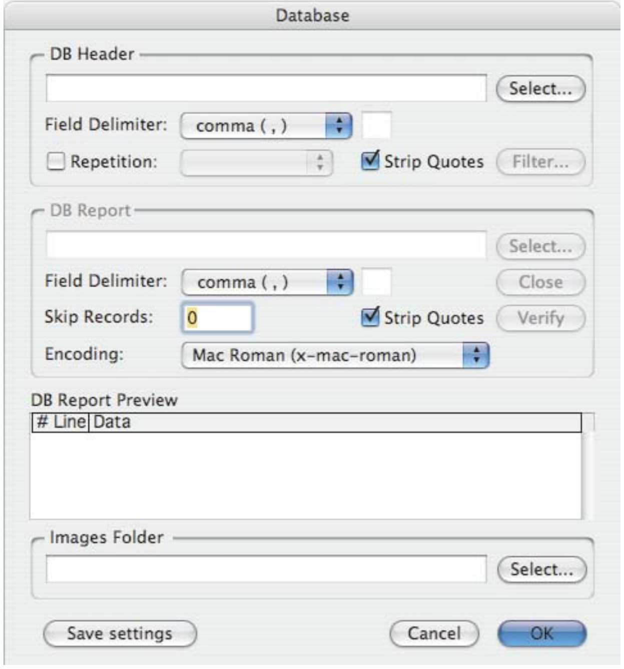
For existing jobs, it is possible to replace the DB header file without losing the assignment of the existing personalization channels to the DB fields, even in the following cases:

- When the new DB header contains new fields.
- When the new DB header does not contain fields that were part of the old job, but with no assignment of Personalization channels.
- When the order of existing fields is changed.
- When you select a DB header. YTD automatically assigns the same file for the DB report.

Note When replacing a DB header file, the original DB field names in the new DB header file should be kept the same as in the original DB header.

To select a database:

1 Click **Yours Truly** and **Database**. The *Database* window appears.



Field Delimiter: comma (,)

☐ Repetition:

☑ Strip Quotes

Field Delimiter: comma (,)

Skip Records: 0

☑ Strip Quotes

Encoding: Mac Roman (x-mac-roman)

Figure 3-2. ***Database* window**

2 Enter the following information in the *Database* window:

a In the DB Header pane, click **Select**. In the browse window that appears, select the DB header file. The header names (that is, column names) are used later in the personalization channel definition.

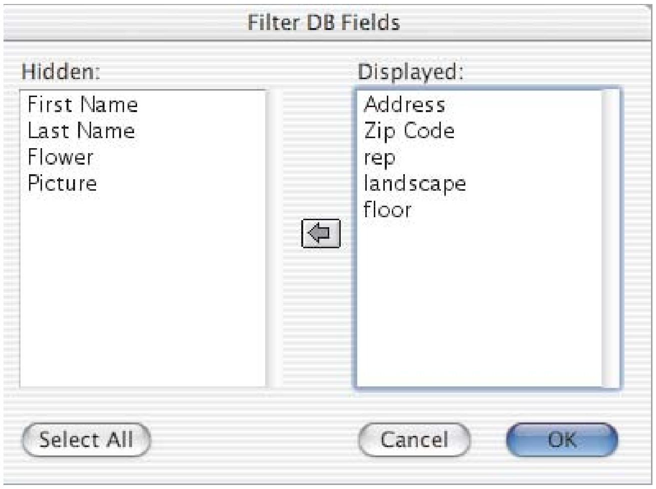
After the DB header file is selected, the same file is automatically selected as the DB Report field. The file contents appear in the DB Report Preview pane.

The DB report file is also saved with your job. There is no need to reassign a DB report file every time you re-open a job.

To remove any DB report file selection, click **Close**.

To choose a different DB report file in the DB Header field, click **Select**. In the browse window that appears, select the DB report file that contains DB records. These are the records that are actually printed.

Note You can simultaneously open several other QuarkXPress layouts assigned to a DB report file.



Hidden:
First Name
Last Name
Flower
Picture

Displayed:
Address
Zip Code
rep
landscape
floor

Figure 3-3. ***Filter DB Fields*** **Window**

b Filtering the database header will help you organize the fields from complex databases by designating the fields that will be used in YTD for that print job. The filtered database header will appear in the repetition field in the *Database* window, in the DB palette, in the *Image Channel Modify* window, and in the *Document Channel Modify* window.

Click the **Filter** button.The *Filter DB Fields* window appears. Move the fields to the *Hidden* or *Displayed* column by selecting the field and using the arrow button. You can click the **Select All** button to select all of the fields in the active column. (see Figure 3-3.)

c In the **Field Delimiter** drop-down menu, select or verify the field delimiter used in the DB header and DB report files.

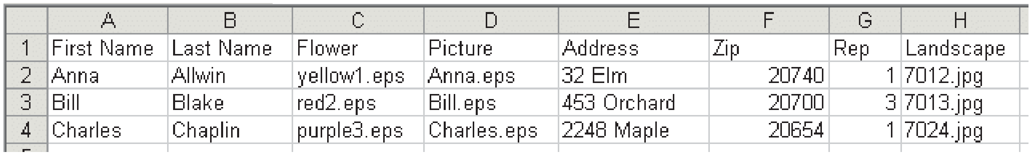
You may also select **Other** from the drop-down menu which enables an extra field in which you may type a relevant character.

d For DB header and DB report files other than Excel, if you want quotes that are part of field values printed, clear the **Strip Quotes** check box.

e If any of the DB records are to be printed more than once, you can indicate this with one of the DB fields. If the DB report file contains a repetition field and you want it to be used, select the **Repetition** check box. Then select the repetition field name from the drop-down menu.

Each line is a DB record, each column is a DB field.

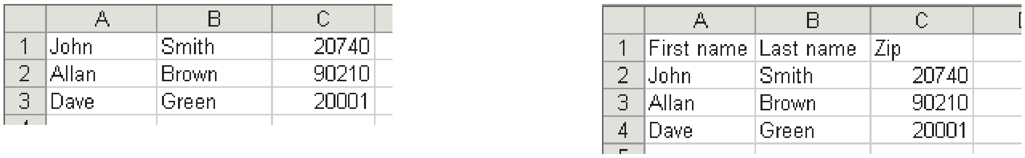
In the following example, the **Rep** field value of **3** for record 3 indicates that record 3 (Bill) will print three times.



	A	B	C	D	E	F	G	H
1	First Name	Last Name	Flower	Picture	Address	Zip	Rep	Landscape
2	Anna	Allwin	yellow1.eps	Anna.eps	32 Elm	20740	1	7012.jpg
3	Bill	Blake	red2.eps	Bill.eps	453 Orchard	20700	3	7013.jpg
4	Charles	Chaplin	purple3.eps	Charles.eps	2248 Maple	20654	1	7024.jpg

Figure 3-4. Example of repetition

f If the DB Report file contains a header record, this record is skipped automatically. If there is no header record, meaning the first record contains actual data, no record is skipped. If more than the one record contains the header information, type the number of such header records in the **Skip Records** field. These header records are skipped during the preview and also during the print of output files which are Job type.



	A	B	C
1	John	Smith	20740
2	Allan	Brown	90210
3	Dave	Green	20001

This DB report contains no header row. The first record contains data.

	A	B	C
1	First name	Last name	Zip
2	John	Smith	20740
3	Allan	Brown	90210
4	Dave	Green	20001

This DB report contains a header row. The first record contains the field names.

Figure 3-5. Example DB reports without and with a header row

g To close the DB report currently selected, click the **Close** button.

h To verify that all the records in the DB report file have the same amount of fields, click **Verify**.

i In the Images Folder pane, click **Select** and select the folder that contains the variable image files. This enables preview of the variable images later.

j Click **Save settings** to save the following parameter values as default: Field Delimiter, Strip Quotes.

k Click **OK**.

You can now define the personalization channels as described in the following sections.

Defining personalization channels

Channels palette

Personalization channel definition is done using the Yours Truly Channels palette after you have designed and prepared your QuarkXPress document and selected a database file.

To start defining personalization channels:

1 In the QuarkXPress menu bar, select **Yours Truly**, **Palettes**, and **Show Main Palettes**. The main palettes, including the Yours Truly Channels palette appear.

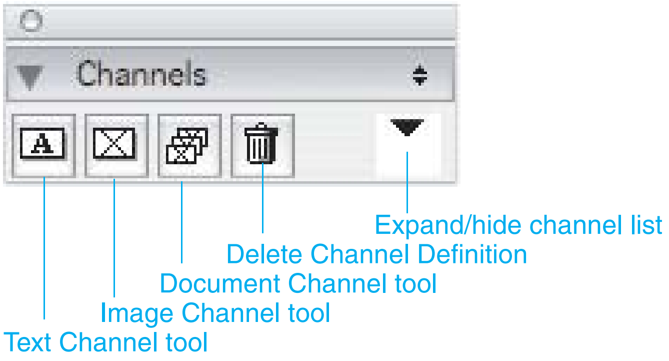
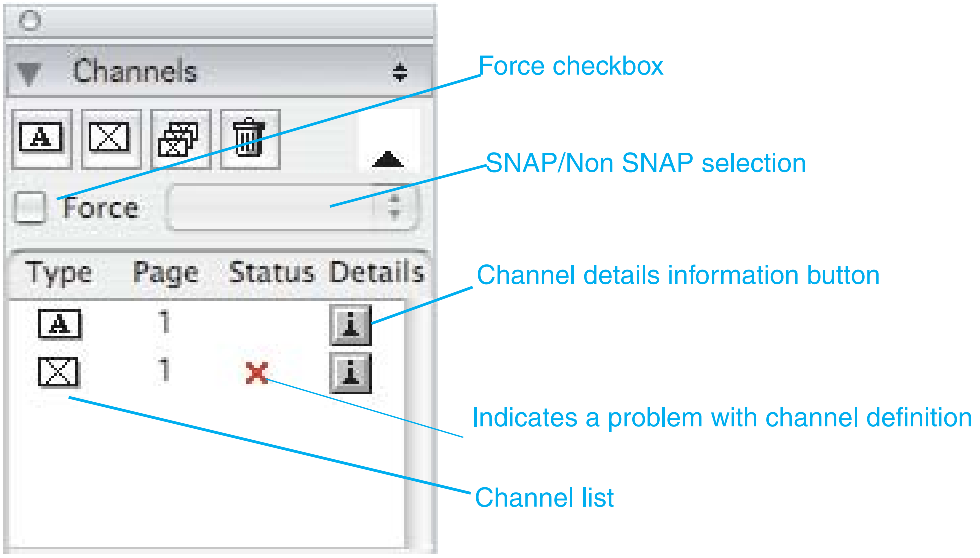


Figure 3-6. Yours Truly Channels palette

2 Click the Expand channel list button. The channel list appears.


Channels
Force
Force checkbox
SNAP/Non SNAP selection
Channel details information button
Indicates a problem with channel definition
Channel list

Type	Page	Status	Details
A	1		i
☒	1	✗	i

Figure 3-7. Yours Truly Channels palette showing channel list

Each separate personalization channel that is defined appears as a different line on the channel list, and contains the following details:

- An icon that identifies the channel type (text, image, or document)
- The channel location in the QuarkXPress document (page)
- The channel's status (a red X appears if there is any problem with the channel definition)
- Additional information about a particular channel can be seen by clicking the ℹ button

Creating template jobs

YTD personalization jobs can be set as:

- SNAP jobs which contain only SNAP channels.
- Non-SNAP jobs which contain only non-SNAP channels.
- Mixed channel jobs which contain both SNAP and non-SNAP personalization channels.

YTD SNAP jobs can produce template type or job type outputs.

Non-SNAP jobs and mixed channel jobs can produce job type outputs only.
(see Table 3-1.)

- Template type outputs do not include an embedded database file or variable data. They are useful for multi-use jobs where the same job may need to be printed more than once, each time using a different DB report file.

  Template type output creation is very fast in comparison to non-template job output creation. This is because personalization channels that are suitable for template type jobs do not need to be processed into PostScript files, and consequently, they are not embedded into the final JLYT or PPML file.

  There are a limited number of channel attributes that are acceptable by a template type job (see detailed list below).

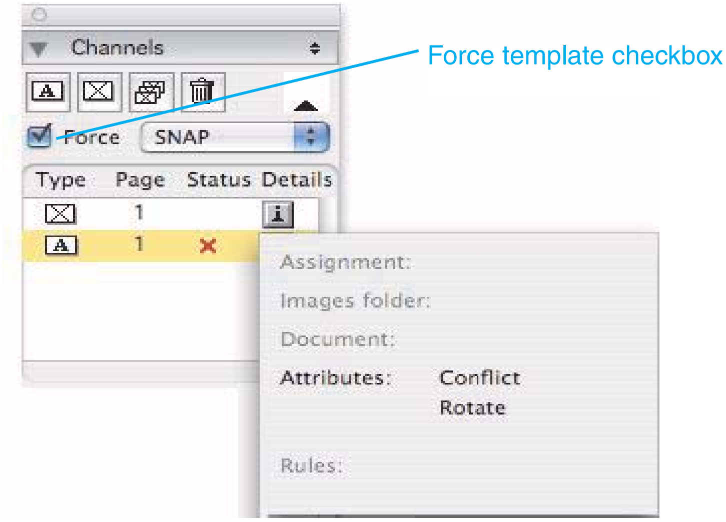
- Job type outputs include an embedded database report file. They can be printed with the embedded DB report data only.

  A job type output may contain a mixture of channels that are acceptable for template jobs and other channels that are not acceptable for template jobs. The channels are processed differently.

  When defining channels for job type output, any attribute available in QuarkXPress can be used in the definition, without any limitation.

To create a template job, check **Force** and select **SNAP** in the Yours Truly Channels palette.

When you use the option to force the job to SNAP, you benefit from a faster workflow process (that is, creation of an output file, transfer to the press, and RIP and loading are all faster).



Assignment:
Images folder:
Document:
Attributes: Conflict
Rotate
Rules:

Figure 3-8. **Force template and details information**

Table 3-1. Job creation options

Channel Type options:	SNAP	Non-SNAP	Mixed Channel
Output file type:	Job or Template	Job	Job

When you check **Force** and select **SNAP**, the different channels can contain the following attributes only:

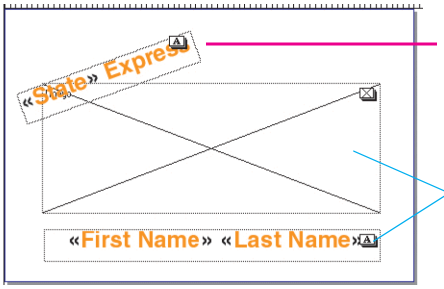
- Text channels:
  - Rectangle text boxes only
  - Font
  - Size (5 to 400 points, whole numbers only)
  - Color
  - Shade
  - Baseline shift
  - Track
  - Leading
  - Alignment
  - Rotate (90^{o}, 180^{o}, 270^{o})
  - Word wrap
  - Purge
  - Overflow cut
- Image channels:
  - Rectangle picture boxes only
  - Rotate (0^{o}, 90^{o}, 180^{o}, 270^{o})
- Document channels:
  - Manual documents only
- Fix element channels:
  - No limitations

When **Force** SNAP is selected and attributes that are not listed above are used, a red X warning sign appears in the channel's Status column (Figure 3-8). Clicking the Channel details information button ℹ opens a window that describes the non-acceptable attributes.

When **Force** is checked, and **Non-SNAP** is selected, all channels result in embedded PostScript files within the output (JLYT) file, regardless of their attributes.

When **Force** is left unchecked, and some channels are suitable for SNAP jobs and some channels are not, the non-SNAP channels result in embedded PostScript files within the output file, while the SNAP channels do not.

When **Force** is left unchecked, any channel attribute is acceptable.



This channel in this job contains non-acceptable SNAP attributes (e.g. rotate) (see above).

Only this non-SNAP channel will result in an embedded PostScript file within the output file.

SNAP channels

Figure 3-9. Job example showing mixed channels

You can define the personalization channels as described in the following sections.

Defining a text channel

Text channel definition consists of defining variable text (QuarkXPress attributes and special YTD attributes), fixed text, and/or a variable index.

1 If the DB Fields palette is not open, click **Yours Truly**, **Palettes**, and **Show DB Fields**. The DB Fields palette appears.

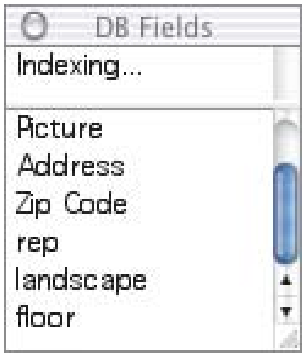


Figure 3-10. DB Fields palette

2 In the main QuarkXPress document, create a text box or text path using any of the QuarkXPress text box or text path tools (except the Tables tool), or select an existing text box.

3 Select the text box/path.

4 In the Yours Truly Tools palette, click the **Text Channel tool** A icon. The text box/path becomes a text channel. An indicator appears in the top right corner of the selected text box.

Figure 3-11. Text channel showing indicator

5 If you want to add variable text (DB fields), do the following:

a Position the cursor in the text box/path where you want the DB field value to appear.

b In the DB Fields palette, select a DB field name. The field name appears in the text box/path.

You can apply QuarkXPress text attributes (font, size, alignment, and so on) in the text box/path to any or all of the text. Regarding field names, an attribute must apply to the entire field name, including the opening and closing chevrons. For example: «First Name».

An example of QuarkXPress text attributes appears in Figure 3-12.

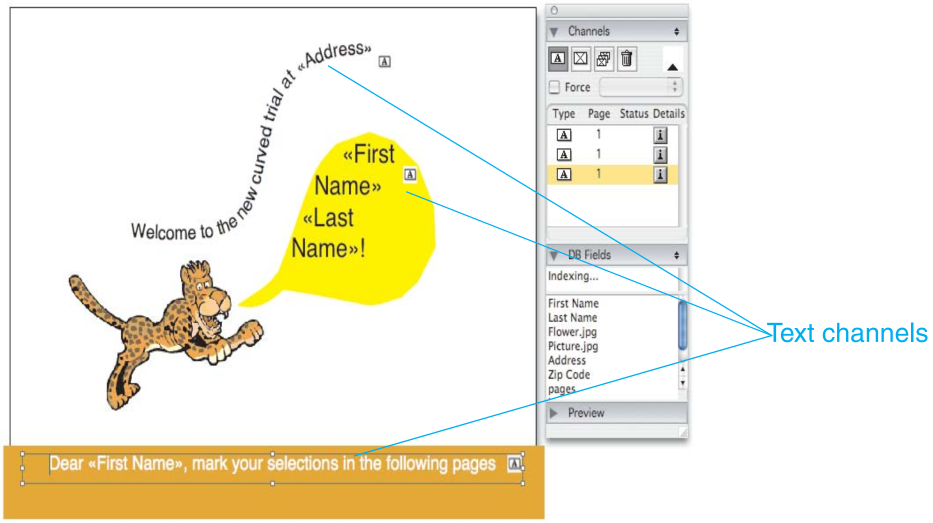


Figure 3-12. Applying QuarkXPress text attributes

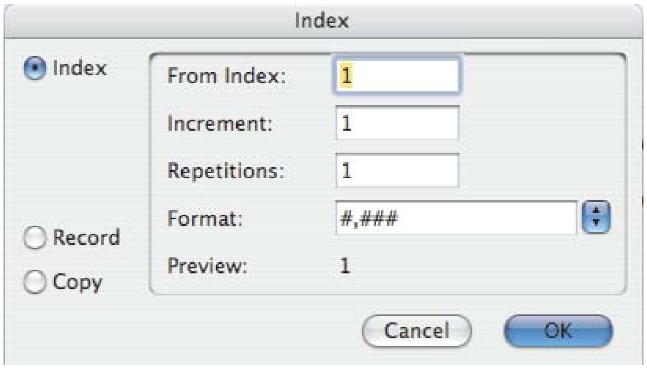
6 If you want to add fixed text, do the following:

a Position the cursor in the text box where you want the text to appear.

b Enter the text.

7 If you want to add an index, do the following:

a In the DB Fields palette, select **Indexing**. The *Index* window appears.



From Index: 1
Increment: 1
Repetitions: 1
Format: #,###
Preview: 1

Figure 3-13. *Index* window

b The **Index** radio button should be selected. If it is not, select it.

c Type the relevant values in the *Index* window.

Field	Description
From Index	Start indexing from this number.
Increment	Increment the index by this value.
Repetitions	Number of times to repeat index. For example: Repetitions = 3 gives 1,1,1,2,2,2,3,3,3,…
Format	Select a display format from the drop-down menu or type your own format, as follows: # = digit 0 = leading/trailing zero , = thousands separator . = decimal point “ ” = text between quotes \ = character after slash
Preview	This shows how the selected **Format** applies to the **From Index** value.

Examples of index formatting:

Format	**Number**	**Result**	
#,###	123 12345	123 12,345	
#	4 1234	4 1234	
000	12 1234	012 1234	
#,000	12 12345	012 12,345	
“Page ”#	123	Page 123	
\P\ #	123	P 123	**Note:** Each slash provides a placeholder for the next character. Here, the second slash holds the place of the space character
“Page ”#“ of 2000”	123	Page 123 of 2000	
#.## (*Custom*)	12 123.4 123.451 123.455	12 123.4 123.45 123.46	
#.000 (*Custom*)	12 123.4	12.000 123.400	

d Click **OK.**

8 If you want to change an index’s settings, do the following:

a In the Preview palette, verify that the **Enable** check box is cleared.

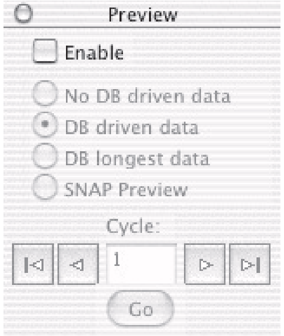


Figure 3-14. Preview palette

b In the text box/path channel, select the entire index field, some of the characters, or place the cursor within the field.

c In the DB Fields palette, select **Indexing**.

d In the *Index* window that appears, change the relevant values.

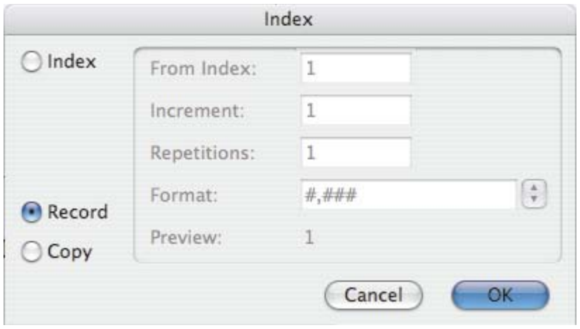
e Click **OK**.

9 To add special counters, do the following:

a In the DB Fields palette, select **Indexing**. The *Index* window appears.

Note The **Record** and **Index** radio buttons are available only in the personalization workflow when the DB header is defined.

b Select either the **Record** or **Copy** radio button. The index fields become unavailable.



From Index: 1

Increment: 1

Repetitions: 1

Format: #,###

Preview: 1

Figure 3-15. *Index* window - Record and Copy counters

The Record counter represents the text channel record number within the DB report file. The Copy counter represents the sequential number of the spreads that contain the text channel.

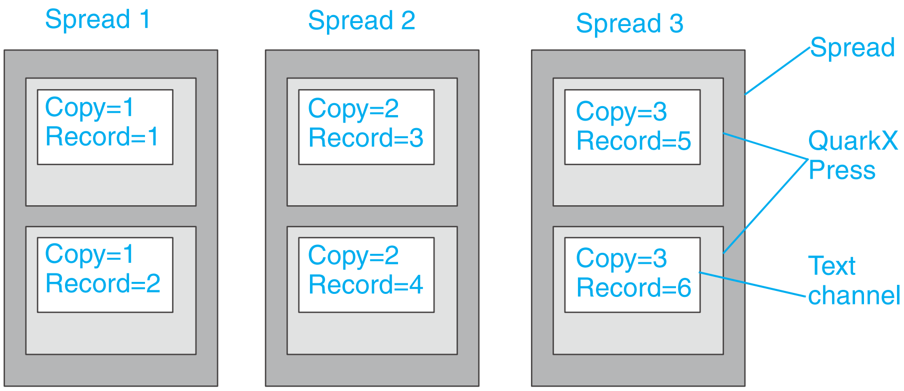


Figure 3-16. Example of Copy and Record counters

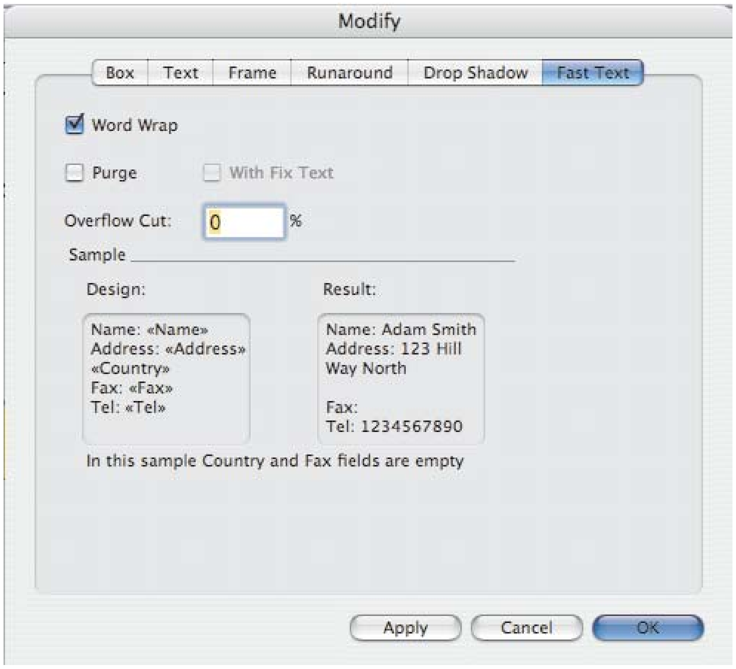
10 When the channel has acceptable Force Template attributes only (see page 26 for a full list of acceptable attributes), you can apply special YTD attributes to the entire text channel:

a Select the text channel.

b In the QuarkXPress menu, click **Item** and **Modify**. The *Modify* window appears.

c Click the **Fast Text** tab.

Note The **Fast Text** tab contents are not available (grayed) when the channel contains non-acceptable template attributes.



☑ Word Wrap

□ Purge □ With Fix Text

Overflow Cut: 0 %

Sample

Design:

Name: «Name»
Address: «Address»
«Country»
Fax: «Fax»
Tel: «Tel»

Result:

Name: Adam Smith
Address: 123 Hill
Way North

Fax:
Tel: 1234567890

In this sample Country and Fax fields are empty

Figure 3-17. Fast Text tab

d The **Word Wrap** check box is selected by default. When selected, the text automatically wraps within the text box.

If this check box is cleared, the text prints without wrapping, and any text beyond the text box border is truncated as show in the example below.

DB Driven Preview	Print / SNAP Preview
☐ **Word Wrap** Name: Adam Smith Address: 123 Hill Way North `Empty` Fax: `Empty` Tel: 123 456 7890	☐ **Word Wrap** Name: Adam Smith Address: 123 Hill W Fax: Tel: 123 456 7890
☑ **Word Wrap** Name: Adam Smith Address: 123 Hill Way North `Empty` Fax: `Empty` Tel: 123 456 7890	☑ **Word Wrap** Name: Adam Smith Address: 123 Hill Way North Fax: Tel: 123 456 7890

For all empty fields on a line, a blank line appears in the print. In the preview, the word `Empty` appears on the empty line. An empty line can cause items to print incorrectly. The **Purge** option deletes the empty line.

Select the **Purge** and/or **With Fix Text** check boxes to correct this phenomenon, as shown in the examples below.

DB Driven Preview	Print / SNAP Preview
☐ **Purge** ☐ **With Fix Text** Name: Adam Smith `Empty` Address: 123 Hill W Fax: `Empty` Tel: 123 456 7890	☐ **Purge** ☐ **With Fix Text** Name: Adam Smith Address: 123 Hill W Fax: Tel: 123 456 7890
☑ **Purge** ☐ **With Fix Text** Name: Adam Smith `Empty` Address: 123 Hill W Fax: `Empty` Tel: 123 456 7890	☑ **Purge** ☐ **With Fix Text** Name: Adam Smith Address: 123 Hill W Fax: Tel: 123 456 7890
☑ **Purge** ☑ **With Fix Text** Name: Adam Smith `Empty` Address: 123 Hill W Fax: `Empty` Tel: 123 456 7890	☑ **Purge** ☑ **With Fix Text** Name: Adam Smith Address: 123 Hill W Tel: 123 456 7890

Note Use the QuarkXPress **Item**, **Modify**, **Text**, and **Vertical Alignment** to determine the positioning of the purged text: top (as in the example above), centered, bottom, or justified.
You can use both **Word Wrap** and **Purge** for the same channel.

The **Overflow Cut** field enables you to reduce the inter-character spacing so that the text fits in the box. Text overflows when the variable text assigned to a text box is larger than the box's capacity. The **Overflow Cut** field's value represents inter-character space reduction in percent. Text that overflows – despite the reduced inter-character spacing – is truncated.

Type the **Overflow Cut** percentage, if needed. It is recommended not to exceed 10 percent. The effects of this parameter do not appear in the preview but in the print.

e Click **OK**.

11 In the Channels palette, verify the channel attributes by checking for the presence of a red X in the Status column. If a red X is present, click on the channel's details information button ℹ, and correct any conflict or error if necessary.

Defining an image channel

This section describes how to insert variable images in your document. The names of the variable images are taken from a DB report file.

1 In the main QuarkXPress document, create a picture box using any of the QuarkXPress picture box tools (except the Tables tool). The picture box will contain variable images.

2 Select the picture box.

3 In the Yours Truly Tools palette, click the **Image Channel tool** icon. ☒

The picture box is now defined as a variable image channel. An indicator appears on the selected picture box.

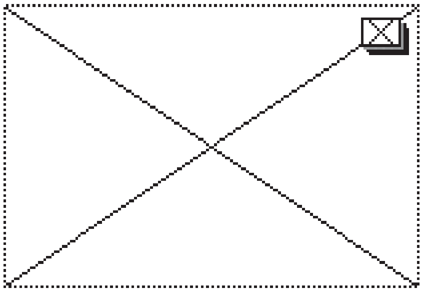
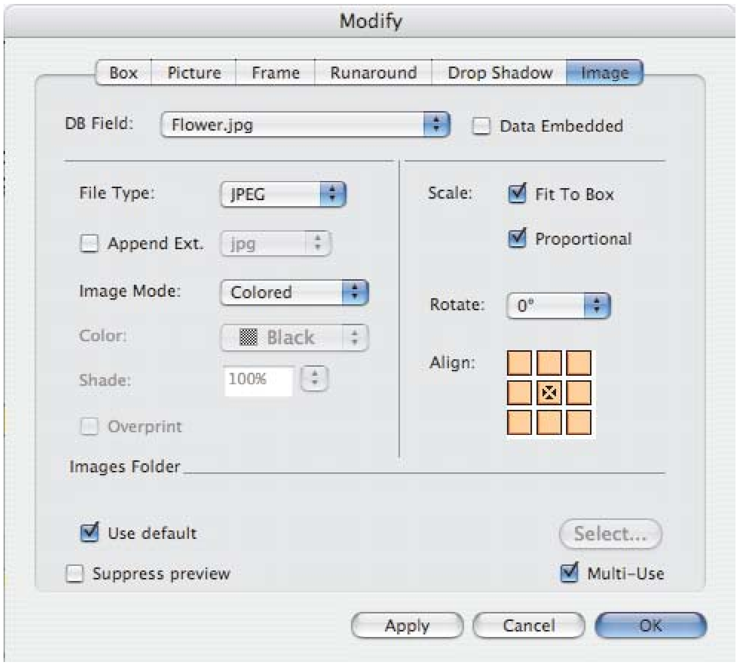


Figure 3-18. Image channel showing indicator

4 In the QuarkXPress menu, click **Item** and **Modify**. The *Modify* window appears.

5 Click the **Image** tab.



DB Field: Flower.jpg
□ Data Embedded

File Type: JPEG
□ Append Ext. jpg
Image Mode: Colored
Color: ▨ Black
Shade: 100%
□ Overprint

Scale: ☑ Fit To Box
☑ Proportional
Rotate: 0°
Align: ☒

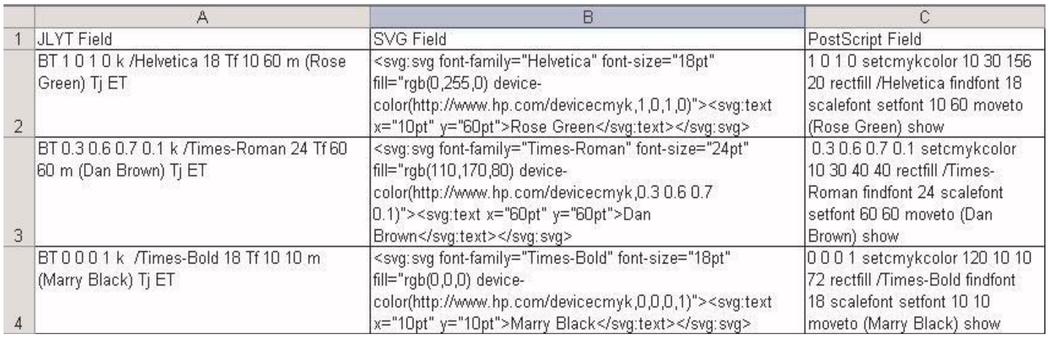
☑ Use default
□ Suppress preview
☑ Multi-Use

Figure 3-19. Image tab

6 In the **DB Field** drop-down menu, select the DB field that represents the names of the images that you want to print in the channel.

You can select the **Data Embedded** checkbox, an advanced option used if the database file contains a field in PS, JLYT-Text, or SVG formats (SVG is available only after selecting the **PPML Enabled** checkbox in the *Preferences* window on the **Ouput** tab, see Chapter 6). When checking **Data Embedded**, the available values in the *File Type* field change to PS, JLYT and SVG. The **DB Field** name selected should be written in that format.

Figure 3-20 shows a sample database file in which: column A is written in JLYT-text format, column B is written in SVG format, and column C is written in PostScript format. The text in the 3 formats is identical, and in addition the PostScript also contains information on drawing the rectangle.



	A	B	C
1	JLYT Field	SVG Field	PostScript Field
2	BT 1 0 1 0 k /Helvetica 18 Tf 10 60 m (Rose Green) Tj ET	<svg:svg font-family="Helvetica" font-size="18pt" fill="rgb(0,255,0) device-color(http://www.hp.com/devicecmyk,1,0,1,0)"><svg:text x="10pt" y="60pt">Rose Green</svg:text></svg:svg>	1 0 1 0 setcmykcolor 10 30 156 20 rectfill /Helvetica findfont 18 scalefont setfont 10 60 moveto (Rose Green) show
3	BT 0.3 0.6 0.7 0.1 k /Times-Roman 24 Tf 60 60 m (Dan Brown) Tj ET	<svg:svg font-family="Times-Roman" font-size="24pt" fill="rgb(110,170,80) device-color(http://www.hp.com/devicecmyk,0.3 0.6 0.7 0.1)"><svg:text x="60pt" y="60pt">Dan Brown</svg:text></svg:svg>	0.3 0.6 0.7 0.1 setcmykcolor 10 30 40 40 rectfill /Times-Roman findfont 24 scalefont setfont 60 60 moveto (Dan Brown) show
4	BT 0 0 0 1 k /Times-Bold 18 Tf 10 10 m (Marry Black) Tj ET	<svg:svg font-family="Times-Bold" font-size="18pt" fill="rgb(0,0,0) device-color(http://www.hp.com/devicecmyk,0,0,0,1)"><svg:text x="10pt" y="10pt">Marry Black</svg:text></svg:svg>	0 0 0 1 setcmykcolor 120 10 10 72 rectfill /Times-Bold findfont 18 scalefont setfont 10 10 moveto (Marry Black) show

Figure 3-20. Sample database

Figure 3-21 shows the output of three records with the PostScript format.

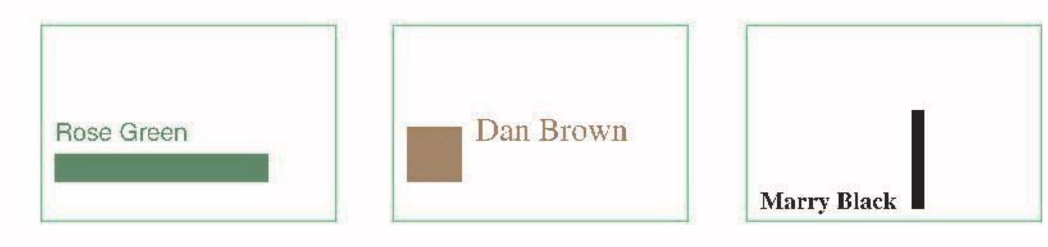


Figure 3-21. Sample output with PostScript format

7 In the **File Type** drop-down menu, select the file type of the image file for the field that you selected. Valid file types are TIFF, JPEG, EPS, PDF, JLYT, and PS.

8 Clear the **Append Ext.** check box if the filename extensions in the DB report file match the actual filename extensions.

Check the **Append Ext.** check box if the actual filenames contain extensions but the filenames in the DB report file do not. In the **Append Ext.** drop-down menu, select the relevant extension name. In this case, the same DB field can be used for a text channel and for an image channel (after the extension is added automatically).

File extensions for the YTD-recognized file types can be added, deleted, or modified. See "Changing image parameters" on page 144.

9 From the **Image Mode** drop-down menu, select one of the following modes:

- **Colored** - This option is selected by default and is most commonly used. Select the color for ordinary printing.

In special cases, for specific coloring issues, you may select one of the following options:

- **Bitmap** - for bitmap images (TIFF and EPS only). If you select Bitmap, the Color and Shade fields become available.

  From the **Shade** drop-down menu, select the shade percentage, or type the shade percentage in the **Shade** field.

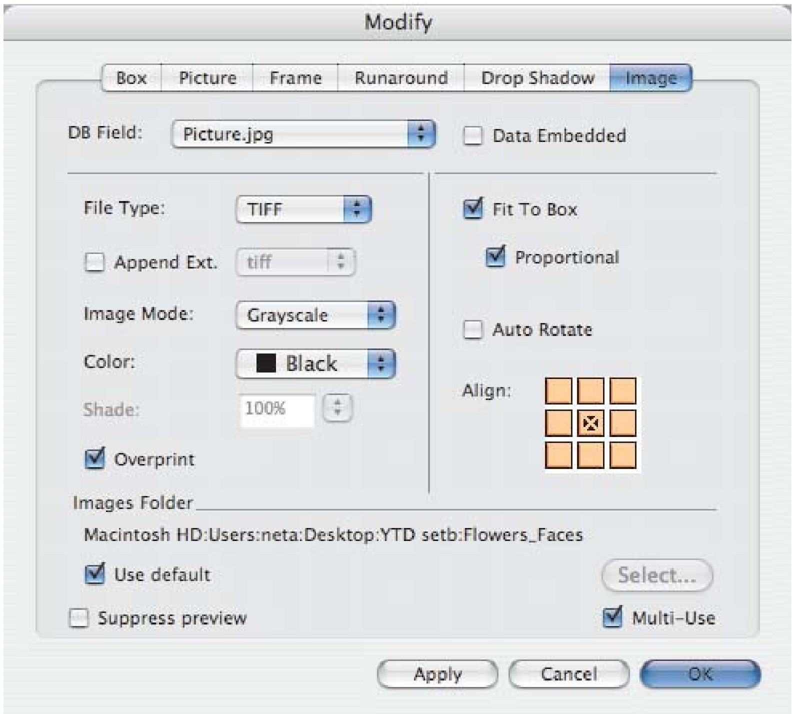
- **Grayscale** - for grayscale images (TIFF, JPEG, and EPS). If you select Grayscale, the Color field becomes available.

  For grayscale images, the following colors are available for coloring the images: cyan, magenta, yellow, black, red, green, and blue. However, grayscale images colored in red, green, or blue are not supported under certain circumstances.

  Selecting Bitmap or Grayscale enables the overprint option.

10 If you select Bitmap or Grayscale, you have the option of overprinting. Overprinting is available for SNAP Image Channels only. The option overprints all color separations not included in the image.

- Enable Force Template in the Channels palette.
- Select the **Overprint** check box in the *Image Modify* window.



DB Field: Picture.jpg

□ Data Embedded

File Type: TIFF

□ Append Ext. tiff

Image Mode: Grayscale

Color: ■ Black

Shade: 100%

☑ Overprint

☑ Fit To Box

☑ Proportional

□ Auto Rotate

Align: ☒

Images Folder

Macintosh HD:Users:neta:Desktop:YTD setb:Flowers_Faces

☑ Use default

Select...

□ Suppress preview

☑ Multi-Use

Figure 3-22. Image tab - Overprint check box

There is no preview available for overprinting. An example of overprinting is shown in Figure 3-23.

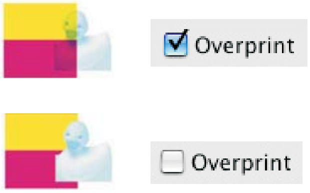
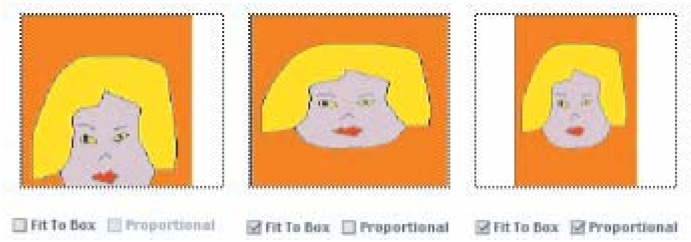


Figure 3-23. Overprinted image

11 If you want the images to fit the picture box completely, check the **Fit To Box** check box. If you check this check box, the **Proportional** check box becomes available.

If you want the images to fit the picture box proportionally, check the **Proportional** check box.

If the **Fit To Box** check box is not checked, the images are inserted in the picture box in their natural size. An example is shown in Figure 3-24.



☐ Fit To Box ☐ Proportional

☑ Fit To Box ☐ Proportional

☑ Fit To Box ☑ Proportional

Figure 3-24. Fit to box

12 To rotate images in the picture box channel, select the relevant angle from the **Rotate** drop-down menu. Select **Auto-rotate** to automatically rotate images 90° clockwise so that their orientation matches the orientation of the picture box.

13 Use the **Align** positioning keys to align the image to nine different positions within the picture box: top/middle/bottom and left/center/right.

14 In the Images Folder pane, check **Use default** to use the images folder defined in the *Database* window, or click **Select** and select the folder that contains the image files. This enables preview of the images, and eventual creation of an output file for a non-template job.

15 If you do not want to preview the images in the main document, select the **Suppress preview** check box.

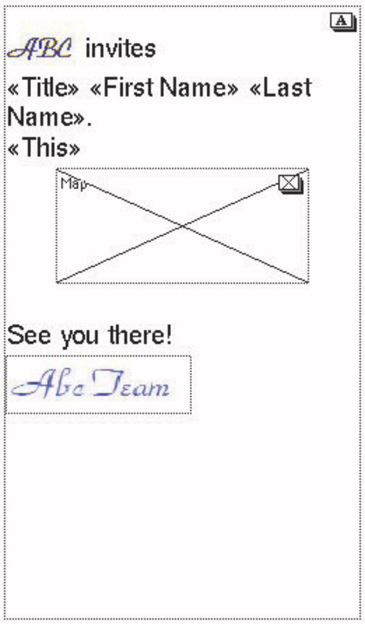
16 Select the **Multi-Use** check box to avoid re-RIPping or reloading images that are used several times in the same DB report or are used with jobs that utilize the same output JLYT file. Clearing this check box forces the images to RIP or load each time they appear in the DB report file or in another job that uses the same output JLYT file.

17 Click **OK**. The **DB Field** name appears in the upper left corner of the image channel. This indicates the completion of a channel assignment.

18 In the Channels palette, verify the channel attributes by checking for the presence of a red X in the Status column. If a red X appears, click on the channel's details information button ℹ , and correct any conflict if necessary.

Defining an anchored image channel within a text channel

This section describes how to anchor an Image Channel or plain image within a Text Channel. Figure 3-25 shows a preview of a Text Channel that contains an Anchored Image Channel and an Anchored Plain Image, in addition to the DB text fields.



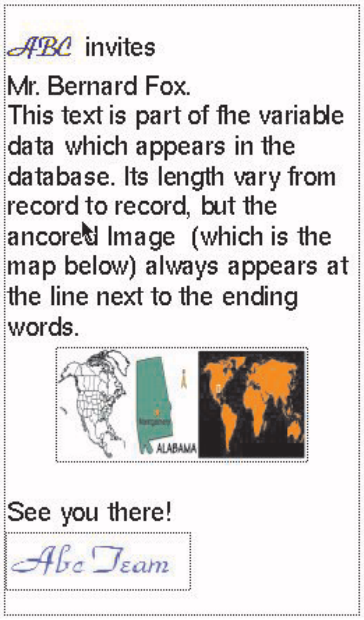
ABC invites
«Title» «First Name» «Last Name».
«This»

Map

See you there!

Abc Team

Anchor sample - Preview disabled



ABC invites
Mr. Bernard Fox.
This text is part of fhe variable data which appears in the database. Its length vary from record to record, but the ancored Image (which is the map below) always appears at the line next to the ending words.

ALABAMA

See you there!

Abc Team

Anchor sample - Preview enabled

Figure 3-25. Anchored Image Channel - Preview disabled and enabled

To define an anchored image channel, use the following steps:

1. Create a Text Channel. For more information, see "Defining a text channel" on page 27.
2. Create an Image Channel (or plain image not defined as a channel). For more information, see "Defining an image channel" on page 34.
3. Copy the image using the QuarkXPress Item tool.
4. Place the cursor at the location that you need to anchor the image.
5. Using the QuarkXPress Content tool, paste the image in the Text Channel.

Defining a document channel

You can insert a document channel into your QuarkXPress document. Three kinds of document channels are supported:

- **Manual document channels** - can be defined when the variable images are taken from a multi-page QuarkXPress document and their names are not in the DB report file.
- **Catalog document channels** - can be defined when the reusable variable data is taken from a DB report file and a multi-page QuarkXPress document.
- **PostScript document channels** - can be defined when the variable images are taken from a multi-page PostScript document and not from the DB report file.

Defining a manual document channel

This section describes how to insert a manual document channel in your document. The variable images are taken from a multi-page QuarkXPress document and their names are not in the DB report file.

1 In the main QuarkXPress document, create a picture box using any of the QuarkXPress picture box tools (except the Tables tool). The picture box will contain variable data.

2 Select the picture box.

In the Yours Truly Tools palette, click the **Document Channel tool** icon. The box is now defined as a document channel. An indicator appears on the selected box.

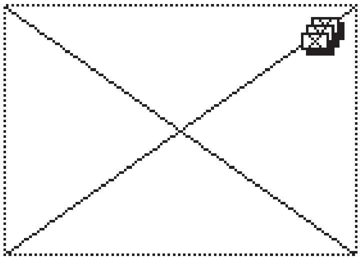
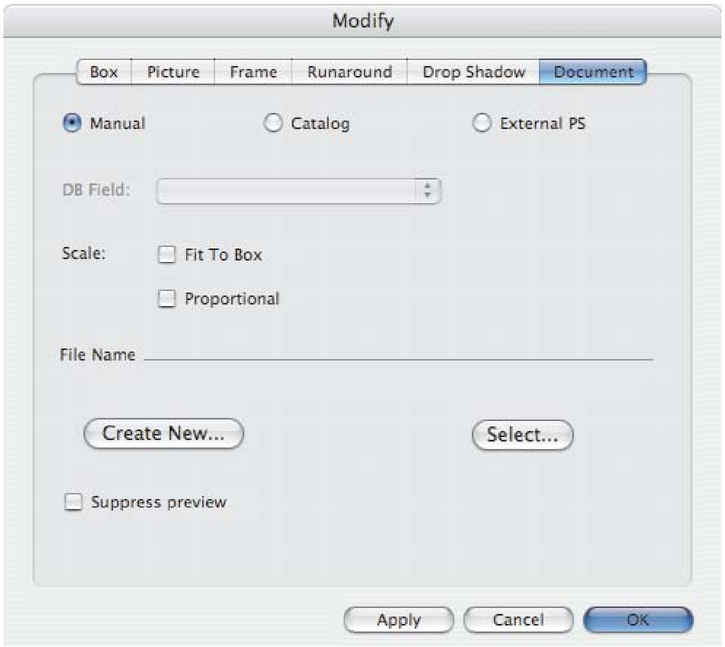


Figure 3-26. Document channel showing indicator

3 In the QuarkXPress menu, click **Item** and **Modify**. The *Modify* window appears.

4 Click the **Document** tab.



● Manual ○ Catalog ○ External PS

DB Field:

Scale: □ Fit To Box

□ Proportional

File Name

□ Suppress preview

Figure 3-27. Document tab

5 Select **Manual**. The manual document channel in the main document will contain variable data from a separate multi-page QuarkXPress document. The variable data from the multi-page document will be visible in the picture box area of the main document.

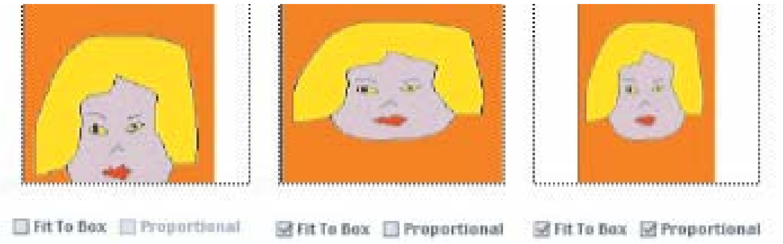
You can create a new multi-page document or select an existing one.

6 To create a new multi-page document, do the following:

a If you want the images to fit in the picture box completely, check the **Fit To Box** check box. If you check this check box, the **Proportional** check box becomes available.

If you want the images to fit in the picture box proportionately, check the **Proportional** check box.

If the **Fit To Box** check box is not checked, the images are inserted in the picture box in their natural size. An example is shown in Figure 3-28.



☐ Fit To Box ☐ Proportional

☑ Fit To Box ☐ Proportional

☑ Fit To Box ☑ Proportional

Figure 3-28. Fit to box

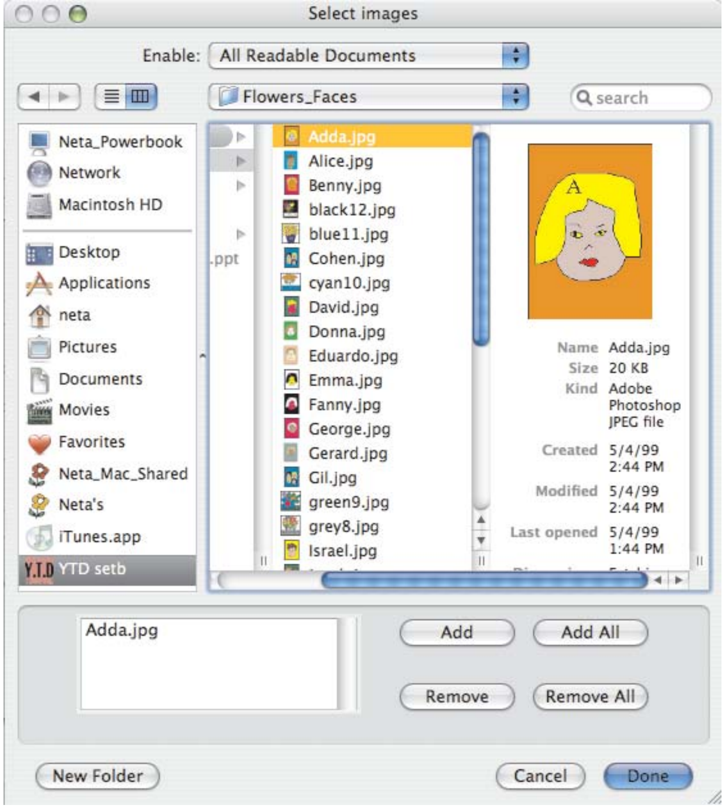
b Click **Create New**.

In the browse window that appears, do the following:

i In the **New Document** field, type the multi-page document name.

ii Select a folder in which to save the document.

iii Click **Save**.
A file selection window appears.



Enable: All Readable Documents

Flowers_Faces

Name Adda.jpg
Size 20 KB
Kind Adobe Photoshop JPEG file
Created 5/4/99 2:44 PM
Modified 5/4/99 2:44 PM
Last opened 5/4/99 1:44 PM

Adda.jpg

Figure 3-29. Image file selection

- **c** Browse to the folder that contains the image files.
- **d** In the upper pane, select the image file that you want to add to the multi-page document.

  Click **Add**. The image file name appears in the lower pane. This pane contains all the image files that will appear in the multi-page document, one image per page, in the order they are printed.

  Add as many image files as necessary, but no more than 2000 due to a QuarkXPress limitation. You can also use **Add All**, **Remove**, and **Remove All**.

  After adding image files from the selected folder, you can select other folders in the upper pane and add images from those folders to the lower pane.
- **e** Click **Done**. The multi-page document is created in the folder you chose earlier. Its name and path appear in the *Modify* window.

7 To select an existing multi-page document, do the following:

- **a** Click **Select**.
- **b** In the browse window that appears, select the multi-page document that contains variable data (images).
- **c** Click **Open**. The multi-page document name and path appear in the *Modify* window.

Note When selecting a multi-page document, **Fit to Box** and **Proportional** options are not available.

The multi-page document is a valid QuarkXPress document and can be edited in QuarkXPress.

8 If you do not want to preview the images in the main document, check the **Suppress preview** check box.

9 Click **OK** to close the *Modify* window. The name of the multi-page QuarkXPress document appears in the upper left corner of the manual channel. This indicates the channel assignment has been changed.

10 In the Channels palette, verify the channel attributes by checking for the presence of a red X in the Status column. If a red X is present, click on the channel's details information button ℹ, and correct any conflict or error if necessary.

Defining a catalog document channel

This section describes how to create a catalog document channel in your document that includes reusable variable images. The reusable variable images are taken from a DB report file and multi-page QuarkXPress document.

1 In the main QuarkXPress document, create a picture box using any of the QuarkXPress picture box tools (except the Tables tool). The picture box will contain variable data.

2 Select the picture box.

3 In the Yours Truly Tools palette, click the **Document Channel tool** icon.

The picture box is now defined as a document channel. An indicator appears on the selected box.

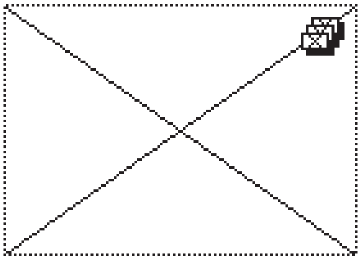
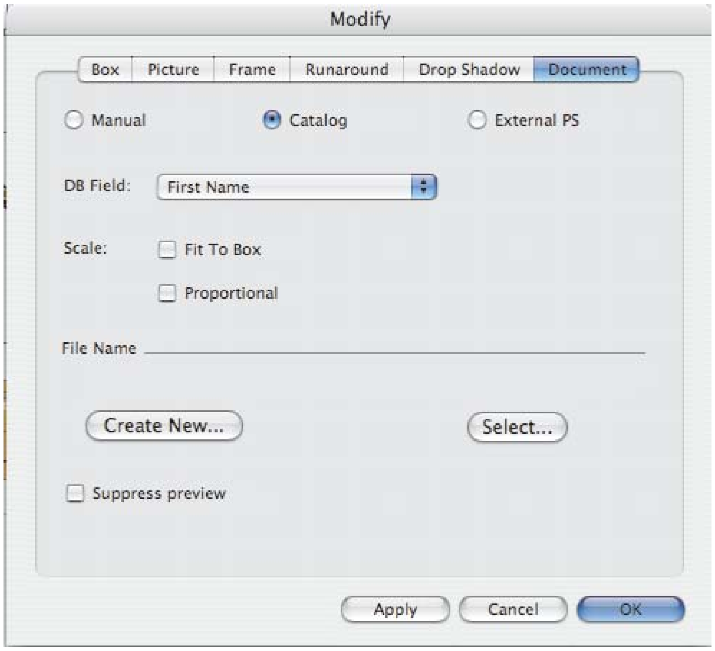


Figure 3-30. Document channel showing indicator

4 In the QuarkXPress menu bar, click **Item** and **Modify**. The *Modify* window appears.

5 Click the **Document** tab.



○ Manual ● Catalog ○ External PS

DB Field: First Name

Scale: □ Fit To Box

□ Proportional

File Name

□ Suppress preview

Figure 3-31. Document tab

6 Select **Catalog**.

7 From the **DB Field** drop-down menu, select the DB field that contains the names of the images that will be inserted into the multi-page document to be defined.

The channel in the main document will contain variable data from a separate multi-page QuarkXPress document. The variable data from the multi-page document will be visible in the picture box area, in the main document.

Note The multi-page document is a valid QuarkXPress document and can be edited in QuarkXPress.

For each page in the multi-page QuarkXPress document, YTD creates a separate PostScript (PS) file. All these PS files are later embedded in the final output JLYT file that you send to the press to print. This is done to enable you to reuse variable data that is referenced by the DB report file, which may contain images and any additional graphic element that may be added while editing the document.

You can create a new multi-page document or select an existing one.

8 To create a new multi-page document, do the following:

a If you want the images to fit into the picture box completely, check the **Fit To Box** check box. If you check this check box, the **Proportional** check box becomes available.

If you want the images to fit into the picture box proportionately, check the **Proportional** check box.

If the **Fit To Box** check box is not checked, images are inserted in the picture box in their natural size as shown in Figure 3-32.

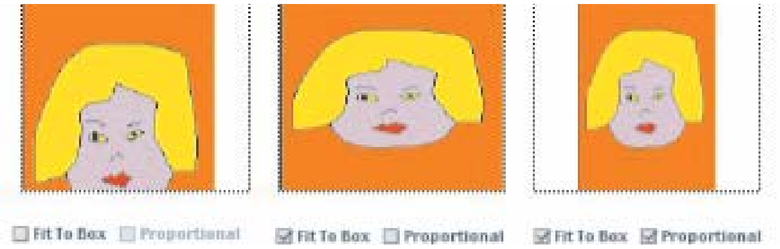

□ Fit To Box □ Proportional
☑ Fit To Box □ Proportional
☑ Fit To Box ☑ Proportional

Figure 3-32. Fit to box

b Click **Create New**.

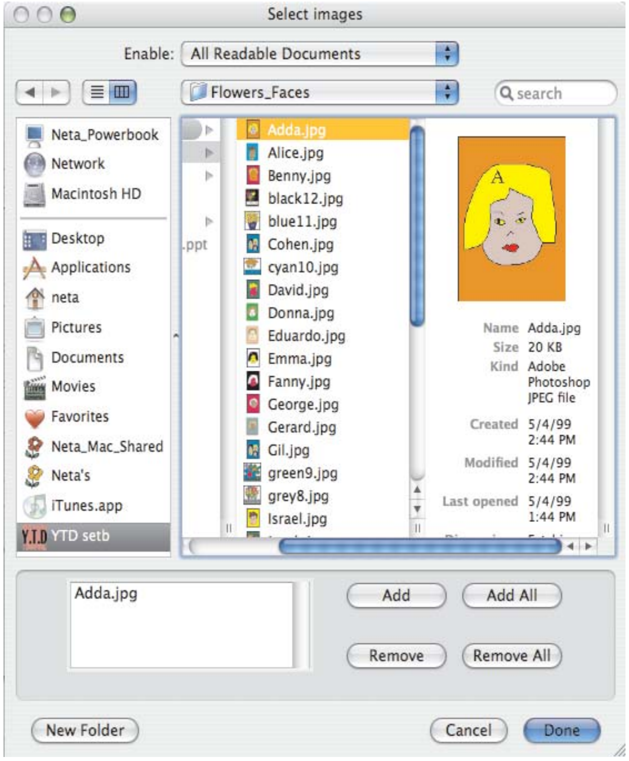
In the browse window that appears, do the following:

i In the **New Document** field, type the multi-page document name.

Note The New Document name must not exceed 24 characters.

ii Select a folder in which to save the document.

iii Click **Save**. A file selection window appears.


Select images

Enable: All Readable Documents

Flowers_Faces

Name Adda.jpg
Size 20 KB
Kind Adobe Photoshop JPEG file
Created 5/4/99 2:44 PM
Modified 5/4/99 2:44 PM
Last opened 5/4/99 1:44 PM

Adda.jpg

Figure 3-33. Image file selection

c Browse to the folder that contains the image files.

d In the upper pane, select the image file that you want to add to the multi-page document.

Click **Add**. The image file name appears in the lower pane. This pane contains all the image files that will appear in the multi-page document, one image per page, in the order their names appear in the DB report file.

Add as many images as necessary, but no more than 2000 due to a QuarkXPress limitation. You can also use **Add All**, **Remove**, and **Remove All**.

After adding image files from the selected folder, you can select other folders in the upper pane and add images from those folders to the lower pane. The various images can be of different types, such as JPEG and EPS.

e Click **Done.** The multi-page document is created in the folder you chose earlier. Its name and path appear in the *Modify* window.

9 To select an existing multi-page document, do the following:

a Click **Select**.

b In the browse window that appears, select the multi-page document that contains variable data (images).

c Click **Open**. The multi-page document name and path appear in the *Modify* window.

10 If you do not want to preview the images in the main document, check the **Suppress preview** check box (see Figure 3-31 on page 44).

11 Click **OK**. The multi-page document name and the DB Field name appear in the upper left corner of the document channel. This indicates that a channel assignment has been made.

Verifying the PostScript names

For each page of the multi-page QuarkXPress document, a PostScript is created. These PostScript file names must be equivalent to the names in the DB Report file that you selected earlier.

To verify the equivalence of the PostScript names, do the following:

1 Click **File** and **Open** to open the multi-page QuarkXPress document.

2 Click **Yours Truly**, **Palettes**, and **Show Naming**.

The PS Cycles Naming palette appears.

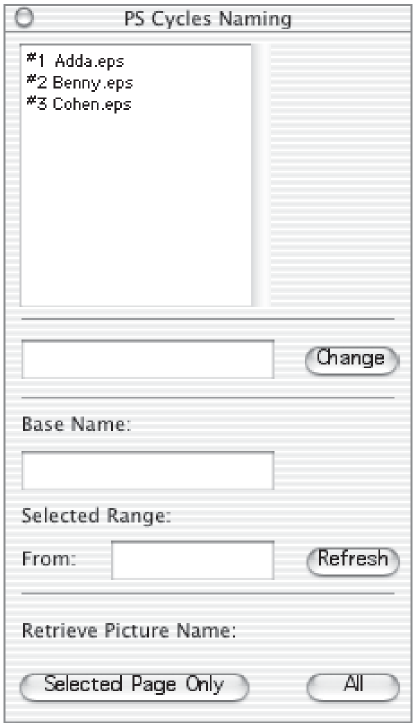
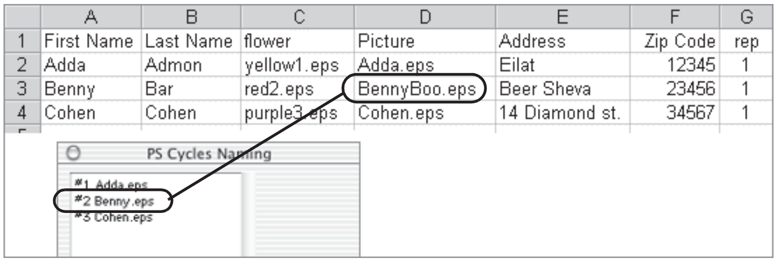


Figure 3-34. PS Cycles Naming palette

The upper pane in the PS Cycles Naming palette contains the list of names of PostScript (PS) files that will soon be created from the pages of the multi-page QuarkXPress document. These names are initially set to the names of the image files.

3 Compare the names in the upper pane to the names in the relevant column of the DB report file. They should be identical.



	A	B	C	D	E	F	G
1	First Name	Last Name	flower	Picture	Address	Zip Code	rep
2	Adda	Admon	yellow1.eps	Adda.eps	Eilat	12345	1
3	Benny	Bar	red2.eps	BennyBoo.eps	Beer Sheva	23456	1
4	Cohen	Cohen	purple3.eps	Cohen.eps	14 Diamond st.	34567	1

Figure 3-35. Example of incorrect file name

If any of the file names do not match, change them in one of the following three ways (see Figure 3-34):

- To change the name to a specific name, do the following:
  - **i** Select the incorrect name in the upper pane.
  - **ii** Type the new name in the text box to the left of the Change button.
  - **iii** Click **Change**.
- If the naming convention is sequential numbering, do the following:
  - **i** In the **Base Name** field, type the naming prefix.
  - **ii** In the **From** field, under Selected Range, type the starting number of the range.
  - **iii** Click **Refresh**. All the names in the upper pane will change accordingly. For example: If the Base Name is `Flower` and the From field is 10, the names change to `Flower_10`, `Flower_11`, `Flower_12`, and so on.
- If the naming convention is by filename, do the following:
  - **i** Select the incorrect name in the upper pane.
  - **ii** Click **Selected Page Only** to change the name to its filename, or click **All** to change all the names to their filenames.

Note

The multi-page QuarkXPress document is a valid QuarkXPress document and can be edited in QuarkXPress. You can add graphics, text, images, and so on, to the document, as needed.

Note

A document catalog channel cannot be part of a template job. Checking **Force template** will result in a red X in the Status column indicating that this is not a valid channel type for this workflow.

4 In the Channels palette, verify the channel attributes by checking for the presence of a red X in the Status column. If a red X is present, click on the channel's details information button ℹ, and correct any conflict or error if necessary.

5 Save and close the multi-page QuarkXPress document.

Defining a PostScript document channel

This section describes how to insert a PostScript document channel in your document. The variable images are taken from a multi-page PostScript document and not from the DB report file.

1 In the main QuarkXPress document, create a picture box using the QuarkXPress rectangle picture box tool only. The picture box will contain variable data.

2 Select the picture box.

In the Yours Truly Tools palette, click the **Document Channel** tool icon. The picture box is now defined as a document channel. An indicator appears on the selected box.

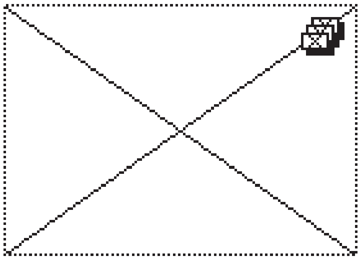
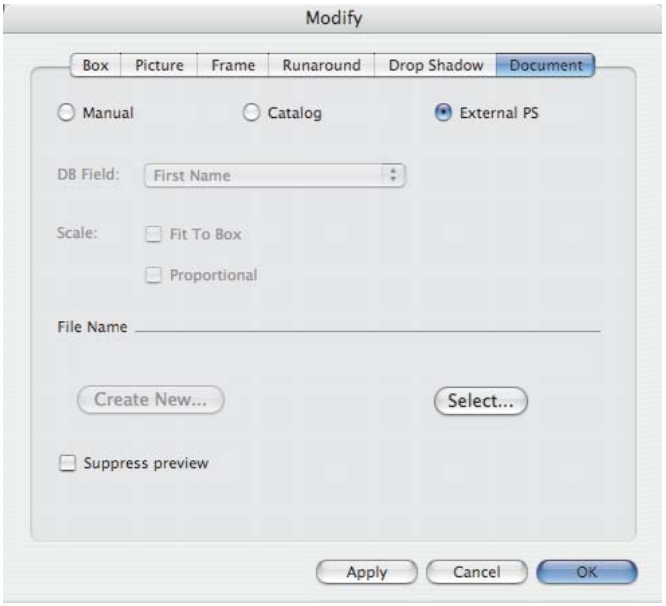


Figure 3-36. Document channel showing indicator

3 In the QuarkXPress menu, click **Item** and **Modify**. The *Modify* window appears.

4 Click the **Document** tab.



○ Manual ○ Catalog ● External PS

DB Field: First Name

Scale: □ Fit To Box

□ Proportional

File Name

□ Suppress preview

Figure 3-37. Document tab

5 Select **External PS**.

The channel in the main document will contain variable data from a separate multi-page PostScript document.

6 Click **Select**.

7 In the browse window that appears, select the multi-page PostScript document that contains variable data (images).

8 Click **Open**. The multi-page PostScript document name and path appear in the *Modify* window.

9 Click **OK**. The name of the multi-page PostScript document appears in the upper left corner of the PostScript channel. This indicates that a channel assignment has been made.

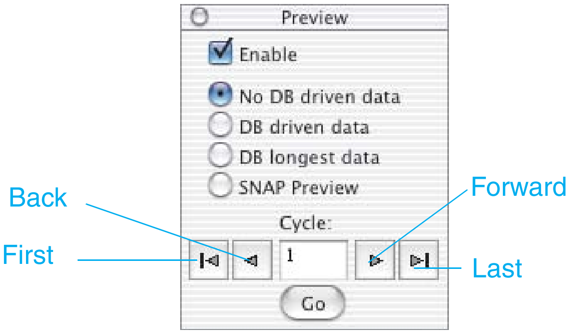
10 In the Channels palette, verify the channel attributes by checking for the presence of a red X in the Status column. If a red X is present, click on the channel's details information button ℹ, and correct any conflict or error if necessary.

Note You cannot preview the variable data in PostScript channels.

Previewing the variable job

To preview the variable text and images in the main document, do the following:

1 If the Preview palette is not opened, click **Yours Truly**, **Palettes**, and **Show Preview**.



Preview

☑ Enable

● No DB driven data
○ DB driven data
○ DB longest data
○ SNAP Preview

Cycle: 1

Figure 3-38. Preview mode enabled

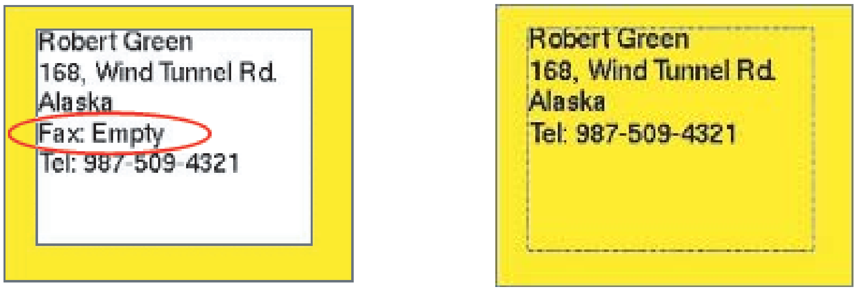
2 In the Preview palette, check **Enable**.

3 To preview the variable images of the personalization channels in the main document, use the forward, back, and **Go** buttons, and the four radio buttons, as follows:

- The **No DB driven data** radio button allows sequential scanning of the images in the images folder that you defined in "Defining an image channel" on page 34 and/or in the multi-page QuarkXPress document that you defined in "Defining a catalog document channel" on page 43 and in "Defining a manual document channel" on page 40. Text channels are not previewed but rather show the variable field names only.
- The **DB driven data job** radio button allows previewing of the variable text and images in the order presented in the DB Report file that you defined during Database definition. The images in a manual channel are previewed in the order that they appear in the multi-page QuarkXPress document that you defined in "Defining a manual document channel" on page 40.
- Selecting the **DB longest data** radio button displays the longest field values for each of the fields in the text channel box. You can thus determine whether the text channel box is large enough to accommodate the largest field values in the current DB report file.

- The **SNAP Preview** radio button allows previewing of the Transparency of channels background, Purge, and Word Wrap. This preview mode shows how the actual text channels will print. For example, true background color will display and “empty” fields will not display. QuarkXPress Tags and Filter XTension should be enabled when using the SNAP Preview.

In Figure 3-39, the empty Fax field displays in the DB driven preview mode as “Empty.” In the SNAP Preview mode, only the true background color and Purge attribute display.



Robert Green
168, Wind Tunnel Rd.
Alaska
Fax: Empty
Tel: 987-509-4321

DB driven data preview mode

Robert Green
168, Wind Tunnel Rd.
Alaska
Tel: 987-509-4321

SNAP Preview mode

Figure 3-39. DB driven data and SNAP Preview modes

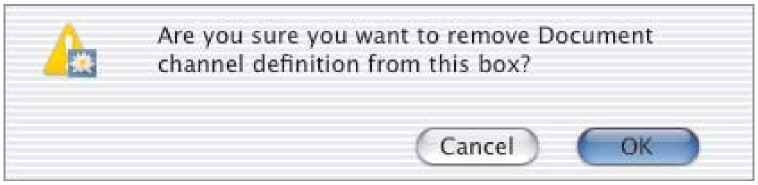
Note Editing channels is not possible when using the SNAP Preview mode.

Cancelling a channel definition

After you define a channel, you may find it necessary to cancel the channel definition from the text/picture box/path, and revert it to a plain QuarkXPress text/picture box/path. To cancel the channel definition, do the following:

1. Click **Yours Truly**, **Palettes**, and **Show Channels**. The Yours Truly Channels palette appears.
2. In the main QuarkXPress document, select the channel.

   The channel’s line is automatically selected in the Channels palette list.
3. Click the **Delete Channel Definition tool** 🗑 button. The following warning message appears.



Are you sure you want to remove Document channel definition from this box?

Figure 3-40. Remove channel warning message

4. Click **OK**.

   The text/picture box/path loses its channel definitions and reverts to a regular QuarkXPress text/picture box/path. Its associated line is deleted from the Channels palette list.

4 Working with YTD rules

This chapter contains the following topics:

- Rules workflows
- Rule structure and syntax
- Importing and editing pre-defined rules
- Creating a new rule
- Assigning rules
- Exporting rules
- Duplicating rules
- Deleting rules
- Syntax elements
- Predefined rules

Rules workflows

Rules can be used to add conditions to fixed text and variable text fields in a text channel, to variable image channels, to layers, or to pages in a personalized copy.

Using rules, you can selectively change the appearance and content of fixed and variable text and variable images based on the rule definitions. You can also selectively print layers or pages in a personalized copy.

A set of predefined image, text, and field rules are supplied with YTD, and can be imported and modified as necessary.

It is recommended that you use the predefined rules. Edit these rules as necessary and apply them to image channels, text fields, and/or fixed text. Creation of new rules from scratch is recommended for experienced users.

Note Rules cannot be applied to document channels.

Two workflows are available for working with rules:

- Import an existing rule, edit it, and apply it to the selected channel. See "Importing and editing pre-defined rules" on page 56.
- Create a new rule and apply it to the selected channel. See "Creating a new rule" on page 61.

The workflows are shown below.

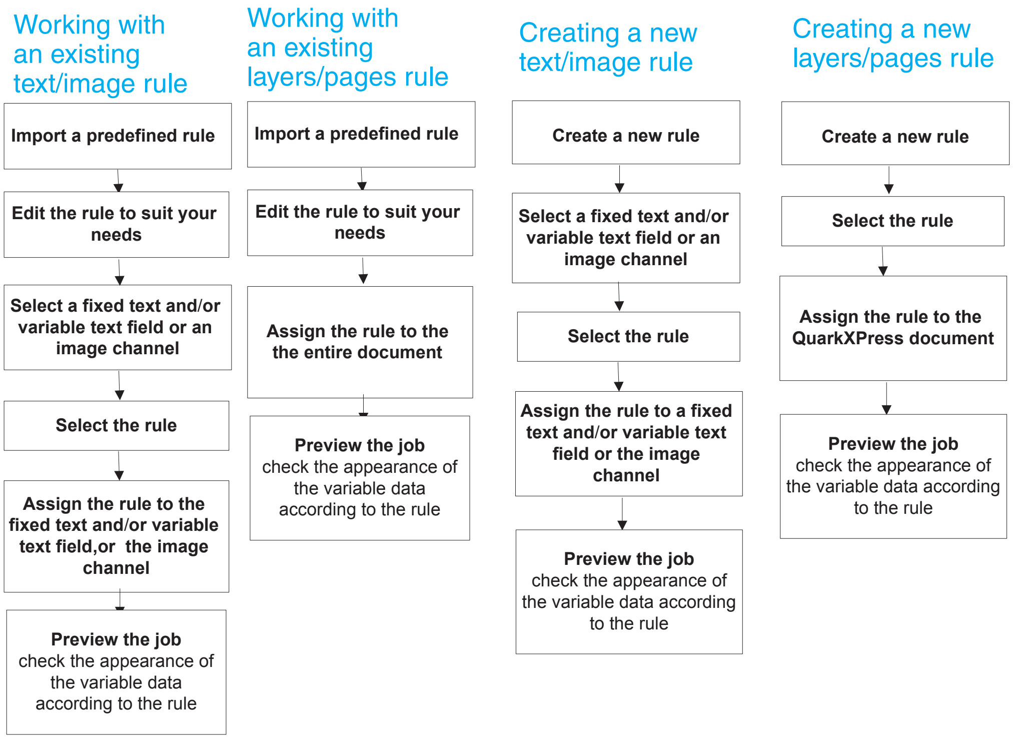

Working with an existing text/image rule
Import a predefined rule
Edit the rule to suit your needs
Select a fixed text and/or variable text field or an image channel
Select the rule
Assign the rule to the fixed text and/or variable text field,or the image channel
Preview the job check the appearance of the variable data according to the rule
Working with an existing layers/pages rule
Import a predefined rule
Edit the rule to suit your needs
Assign the rule to the the entire document
Preview the job check the appearance of the variable data according to the rule
Creating a new text/image rule
Create a new rule
Select a fixed text and/or variable text field or an image channel
Select the rule
Assign the rule to a fixed text and/or variable text field or the image channel
Preview the job check the appearance of the variable data according to the rule
Creating a new layers/pages rule
Create a new rule
Select the rule
Assign the rule to the QuarkXPress document
Preview the job check the appearance of the variable data according to the rule

Figure 4-1. Rule application workflow

Rule structure and syntax

Rules used in YTD are expressed as verbal sentences called expressions. They take the form of conditional statements such as:

IF (logical test), THEN (value if true), ELSE (value if false)

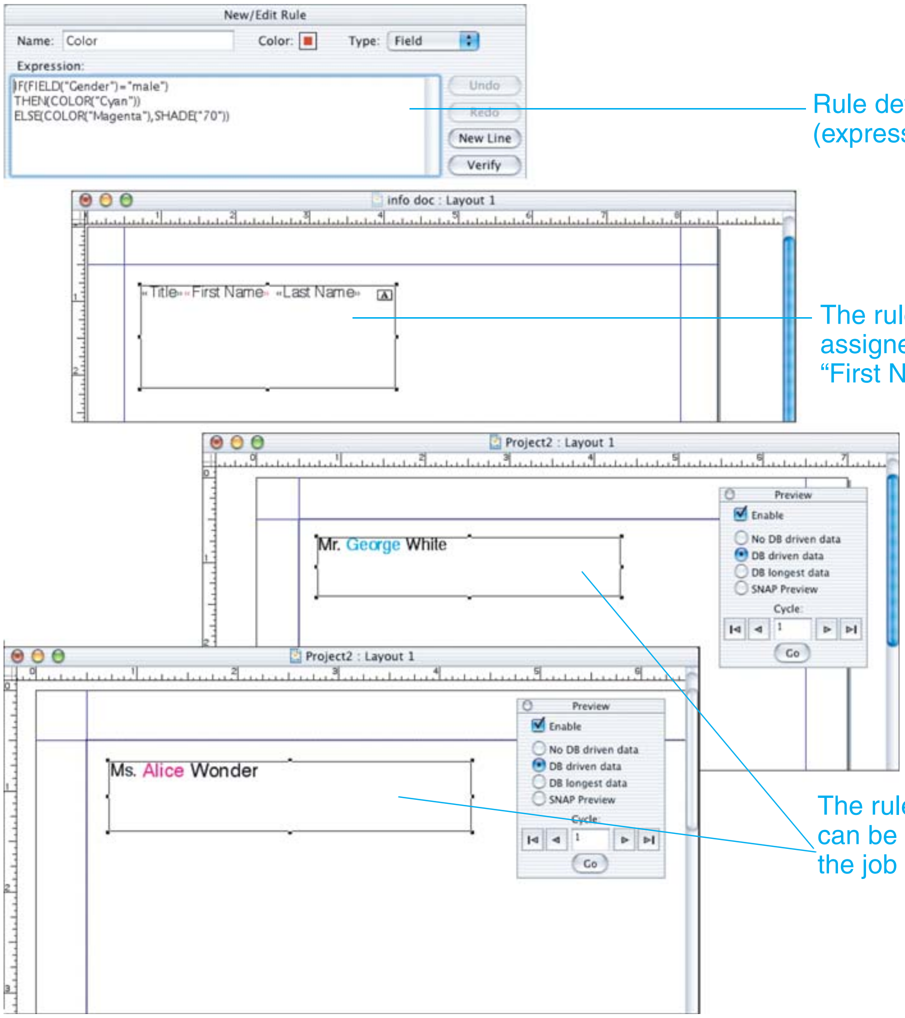
This basic conditional statement can be expanded to include more complex conditions and functions.

The basic functions, operators, and attributes used in rule syntax are described under "Syntax elements" on page 77.

Example

This example uses the Color rule, which is a text or field rule that can be assigned to any text. In the example shown below, the rule is assigned to the “First Name” field.

This rule reads the database field named “Gender”. If the field value is “male”, then the assigned field on the text channel (First Name in the example) is printed in cyan. If the “Gender” field value is not “male”, the assigned field is printed in magenta. The same rule can be assigned to more than one text field and to fixed text as well.



New/Edit Rule

Name: Color Color: ■ Type: Field

Expression:

```
IF(FIELD("Gender")="male")
THEN(COLOR("Cyan"))
ELSE(COLOR("Magenta"),SHADE("70"))
```

info doc : Layout 1

«Title» «First Name» «Last Name»

Project2 : Layout 1

Mr. George White

Project2 : Layout 1

Ms. Alice Wonder

Preview: ☑ Enable ○ No DB driven data ● DB driven data ○ DB longest data ○ SNAP Preview Cycle: 1

Rule details (expression)

The rule is assigned to the “First Name” field

The rule results can be seen in the job preview

Figure 4-2. Rule application example

Importing and editing pre-defined rules

This section describes how to import and edit predefined rules.

Importing rules

Import rules from the set of predefined rules that are supplied with the YTD software. You can then modify these rules as necessary.

The predefined rules are provided in the *MAC hard disk, MAC User, Library, Preferences, Quark, QuarkXPress 7, Yours Truly, Rule* folder. Clicking on the **Import rule** button opens the folder in which the rules are located. See below for details.

Rule Types

Image Rule

Assign an Image Rule to an image channel. It can affect the selection of the image according to the rule definition. An image can be a specific image named in the rule or an image taken from the database from a specific column.

Field Rule

Assign a Field Rule to database fields. The Field Rule can control the text attributes (such as color, size, font type) and text content.

Text Rule

Assign a Text Rule to either database fields or to fixed data. The Text Rule will be assigned to any selected text according to the user selection. A Text Rule can control only text attributes and not text content.

Pages Rule

For PPML workflow only. Assign a Pages Rule to the entire QuarkXPress document (layout). The Pages Rule is used to select specific QuarkXPress pages from a layout as a rule condition. The Pages Rule can control the number and order of pages in each personalization copy.

Layers Rule

For PPML workflow only. Assign a Layers Rule to the entire QuarkXPress document (layout). The Layers Rule can control the appearance of the different layers in each personalization copy.

To import a rule for use with your text or image channel:

1 In the QuarkXPress menu bar, select **Yours Truly**, **Palettes**, and select **Show Rules**. The Rules palette appears.

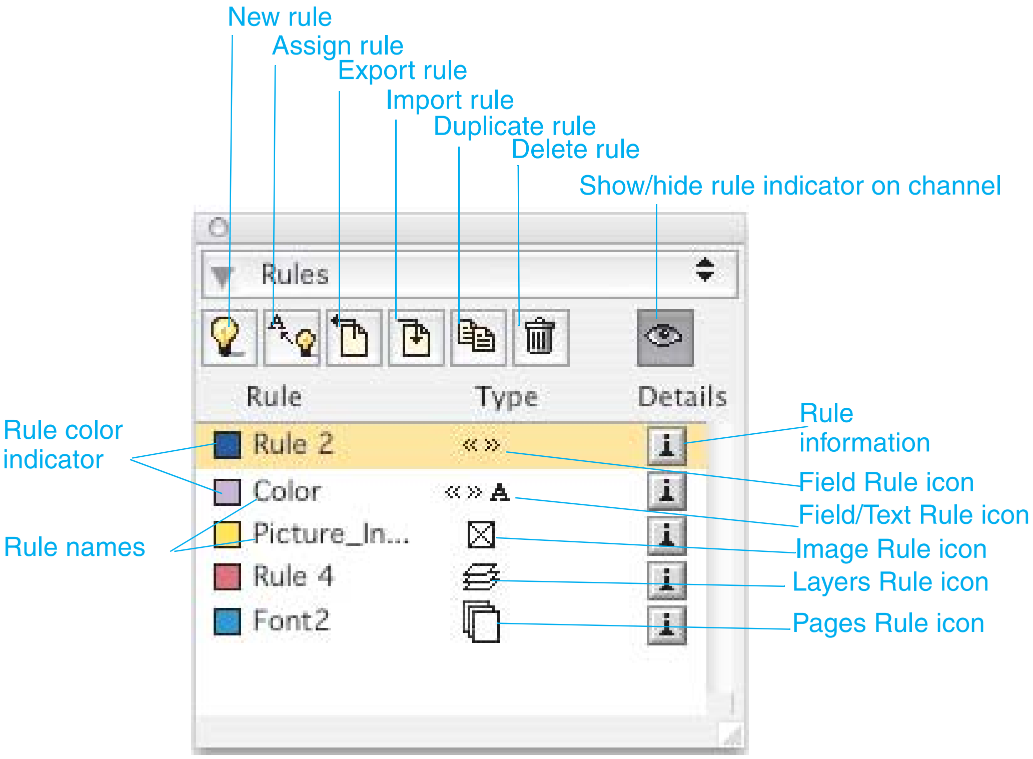


Figure 4-3. Rules palette

2 On the Rules palette, click the **Import** button . The Rule list opens.

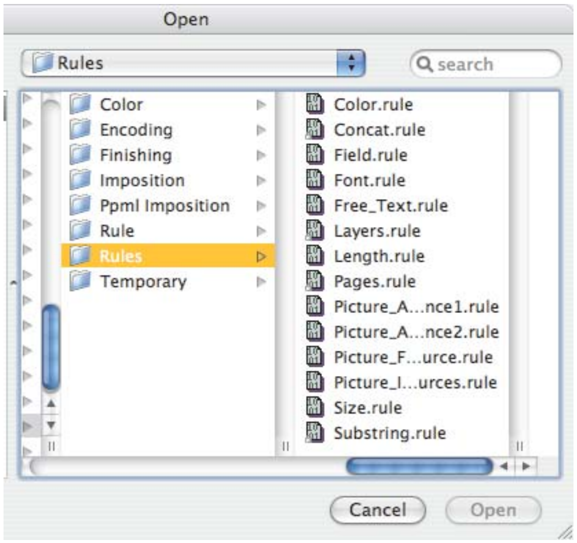


Figure 4-4. Rule list

3 Select a rule that suits your needs from the list (See the pre-defined rules in "Predefined Image Rules" on page 82, "Predefined Text/Field Rules" on page 80, "Predefined Pages Rule" on page 83, and "Predefined Layers Rule" on page 83).

4 Click **Open**. The rule appears in the Rules palette list. Each imported or new rule is automatically assigned a unique indicator color.

5 Rule types are identified in the Type column as follows (Figure 4-3.):

- Field rules - << >>
- Text rules - “A”
- Image Rules - ☒
- Rules suitable for both fields and text - <<>>A
- Page rules - ⧉
- Layer rules - ≋

You can click the ℹ button to see the rule’s expression.

Editing a rule

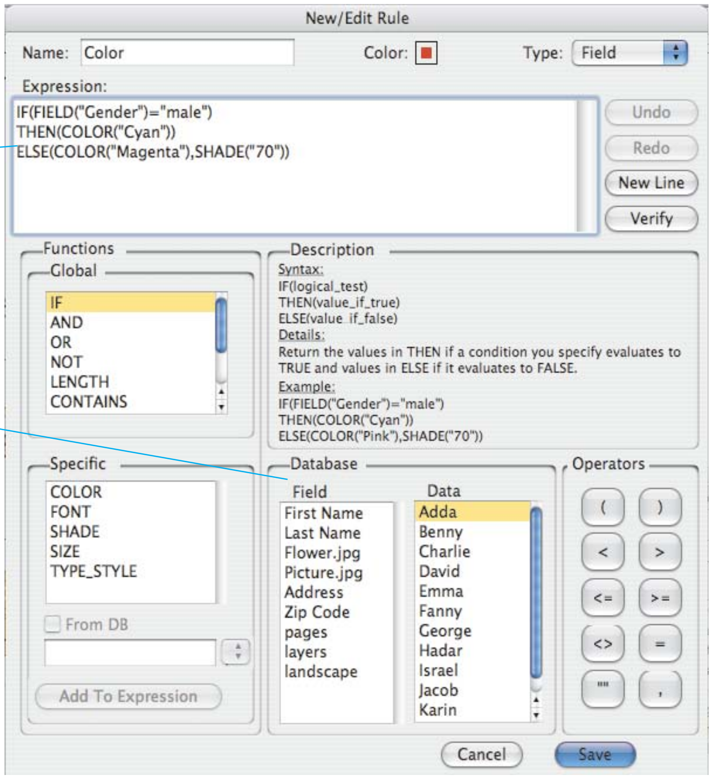
The pre-defined rule expression contains DB field names and values that do not necessarily match your current DB file header and contents.

Edit the rule in order to replace the DB fields with those found in your current DB file, and to enter any other relevant values instead of, or in addition to, those that appear in the imported rule.

The rule editing procedure is illustrated here using the Color Field/Text Rule.

To edit an imported rule:

1 On the Rules palette, double-click on a rule, the *New/Edit Rule* window opens.



Name: Color Color: 🟥 Type: Field

Expression:

```
IF(FIELD("Gender")="male")
THEN(COLOR("Cyan"))
ELSE(COLOR("Magenta"),SHADE("70"))
```

Syntax:
IF(logical_test)
THEN(value_if_true)
ELSE(value_if_false)
Details:
Return the values in THEN if a condition you specify evaluates to TRUE and values in ELSE if it evaluates to FALSE.
Example:
IF(FIELD("Gender")="male")
THEN(COLOR("Cyan"))
ELSE(COLOR("Pink"),SHADE("70"))

Original expression of pre-defined rule

DB header field names and data entries from the current job

Figure 4-5. New/Edit Rule window

2 You can change the rule name by typing a name for the rule in the Name field (Color in the example above).

3 You can change the rule color indicator by clicking the **Color** button, and selecting a color from the color palette.

4 In the Type drop-down menu, make sure that your rule is of the correct type (view the Type Field and see that both Field and Text types are checked for the Color Rule).

5 Edit the "IF" statement:

a In the Expression field, double-click on the DB field name ("Gender" in the example above). The DB field name becomes selected.

b In the Database pane, Field list, double-click on a database field to insert it into the expression. The name of the database field appears in the expression (replacing "Gender" in the example).

c If you need to modify the operator, double-click on the existing operator to select it (the = sign in the example above). In the Operators pane, click on the appropriate button in the list. The new operator replaces the existing operator in the expression.

d In the Expression field, select the field value ("male" in the example above), and double-click a value from the Data pane. The selected value replaces the original value in the expression.

6 Edit the "Then" expression. You can do one of the following:

- Use the existing Expression attribute, and change only its value (see step a below).
- Replace the Expression attribute and its value (see step b below).
- Add attributes to the expression (see step c below).

Note You can verify the syntax of the expression at any time in the editing process by clicking **Verify**. You can undo the last edit action by clicking **Undo**, keep clicking **Undo** to undo previous changes. You can redo the last edit action by clicking **Redo,** and keep clicking **Redo** to redo subsequent changes.

a To modify the first attribute value (in the example above, the Cyan for the COLOR attribute):

i Double-click on the attribute value in the expression to select it (Cyan in the example above).

ii In the Specific pane list, click on the desired attribute (Color in the example). The attribute values list becomes available.

iii Select the desired attribute value from the drop-down list.

iv Click **Add to Expression**. A new attribute value replaces the original attribute value in the expression.

b To replace the attribute and its value in the expression (COLOR("Cyan") in the example above):

i Select the attribute and its value in the expression. In the example, select COLOR("Cyan").

ii In the Specific pane list, select a new attribute. The attribute values become active.

iii Click the drop-down list and select an attribute value.

iv Click **Add to Expression**. A new attribute and value replace the selected attribute and its value.

c To add an attribute and value to the first attribute line (after THEN(COLOR(“Cyan”) in the example above):

i Place the cursor before the last parenthesis of the line.

ii In the Operators pane, click the comma button. A comma appears between the parentheses.

iii In the Specific pane list, select an attribute to add to the expression. The attribute values become active.

iv Click the drop-down list and select an attribute value.

v Click **Add to Expression**. The attribute and its value appear following the comma and the existing first attribute and its value in the expression.

7 Edit the “Else” statement by repeating the actions described in Step 6, as necessary.

8 You can increase the complexity of the rule:

a To add a new line to the expression, click **New Line**, and edit the new line.

b To use a function other than IF, click the function name in the Functions pane. The function appears at the cursor location in the Expression field.

Note When you select a function in the Functions pane, syntax details and an example of the function appear in the Description pane to guide you in creating your expression.

9 When you have finished editing your expression, click **Verify**. The function is verified. You are notified of any errors in the expression.

10 When the rule is completed without errors, click **Save.** The rule is verified and saved under the name provided in the Name field.

Creating a new rule

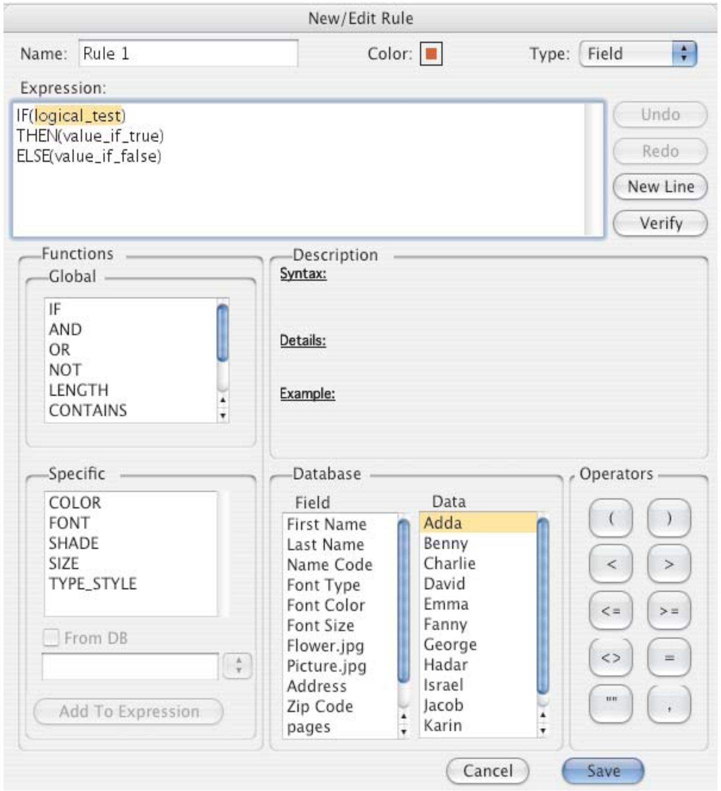
Creating a new field rule

The rule creation process described below is illustrated with the creation of a new field rule using the LENGTH function. This rule can be used to prevent variable text from overrunning the limits of the text channel box and of being truncated. Using the LENGTH rule, the font of the text is reduced based on the number of characters in the text string.

New users should gain experience by editing existing predefined rules to match their needs (see "Editing a rule" on page 71).

To create a new rule:

1 Click the **New rule** button, opening the *New/Edit Rule* window.



Name: Rule 1 Color: ■ Type: Field

Expression:

```
IF(logical_test)
THEN(value_if_true)
ELSE(value_if_false)
```

Figure 4-6. ***New/Edit Rule* window**

2 In the Name field, type a name for the rule.

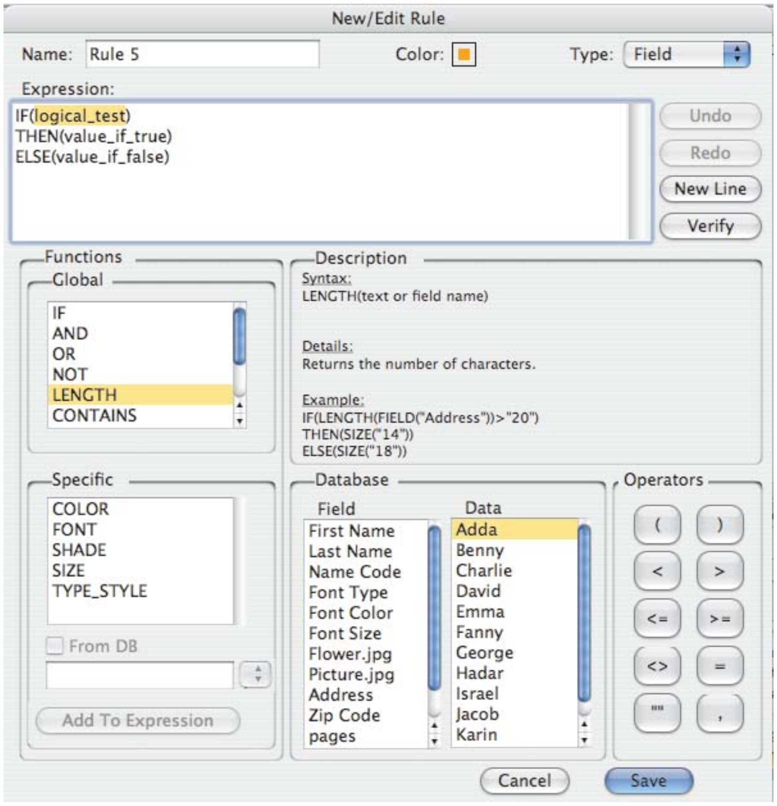
3 You can change the rule color by clicking the **Color** button, and selecting a color from the color palette.

4 In the Type drop-down menu, make sure the rule you use is of the field type.

5 Edit the “IF” statement:

   a In the Expression field, select the **logical_test** expression.

   b In the Functions pane, select the desired function (LENGTH in the example). The function syntax, details, and example are shown on the right of the Functions pane (Figure 4-10).



New/Edit Rule

Name: Rule 5

Type: Field

Expression:

```
IF(logical_test)
THEN(value_if_true)
ELSE(value_if_false)
```

Description

Syntax:
LENGTH(text or field name)

Details:
Returns the number of characters.

Example:

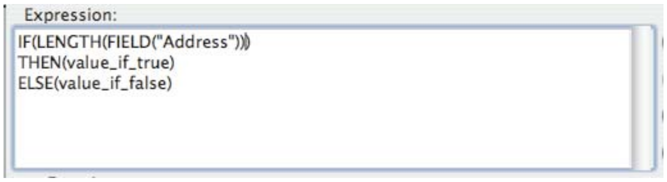
```
IF(LENGTH(FIELD("Address"))>"20")
THEN(SIZE("14"))
ELSE(SIZE("18"))
```

Figure 4-7. Editing the *If* statement

c Double-click on the desired function. The function syntax appears in the expression.

d Modify the function as necessary, using the function syntax example shown in the Functions pane:

i In the Database pane, Field list, double-click on the required database field in order to insert into the expression (Address, for example). The name of the database field appears in the expression.



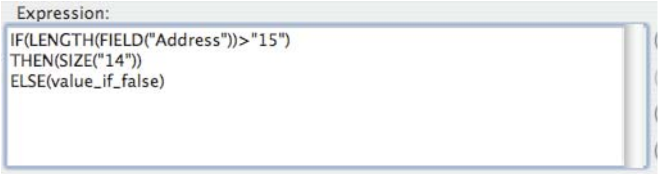
Expression:

```
IF(LENGTH(FIELD("Address")))
THEN(value_if_true)
ELSE(value_if_false)
```

Figure 4-8. Inserting the database field into the expression

ii Place the cursor to the left of the last parenthesis and add an operator by clicking the appropriate button in the Operators list (> for example). The selected operator appears after the database field name in the expression.

iii Type the required logical test value following the operator in the expression ("15", including the quotation marks, which provides the number of characters in the selected field, in the example).



```
IF(LENGTH(FIELD("Address"))>"15")
THEN(SIZE("14"))
ELSE(value_if_false)
```

Figure 4-9. Inserting logical test value into the expression

6 Edit the "Then" statement:

a Select the **value_if_true** expression.

b In the Specific list, select an attribute to insert into the expression (SIZE for example). The attribute values become active.

c Click on the attribute values drop-down list and select an attribute value (14, which represents the font size, in the example).

d Click **Add to Expression**. The attribute and its value appear in the expression.

7 Edit the "Else" statement:

a In the Expression field, place the cursor in the **value_if_false** statement in the expression and select it.

b Click on the attribute values drop-down list and select an attribute value (18, which represents the font size, in the example).

c Click **Add to Expression**. The attribute and its value appear in the expression.

Note You can verify the syntax of the expression at any time in the editing process by clicking **Verify**. You can undo the last edit action by clicking **Undo**. You can redo the last edit action by clicking **Redo**.

Undo and **Redo** can be clicked back one step at a time until the last saved expression.

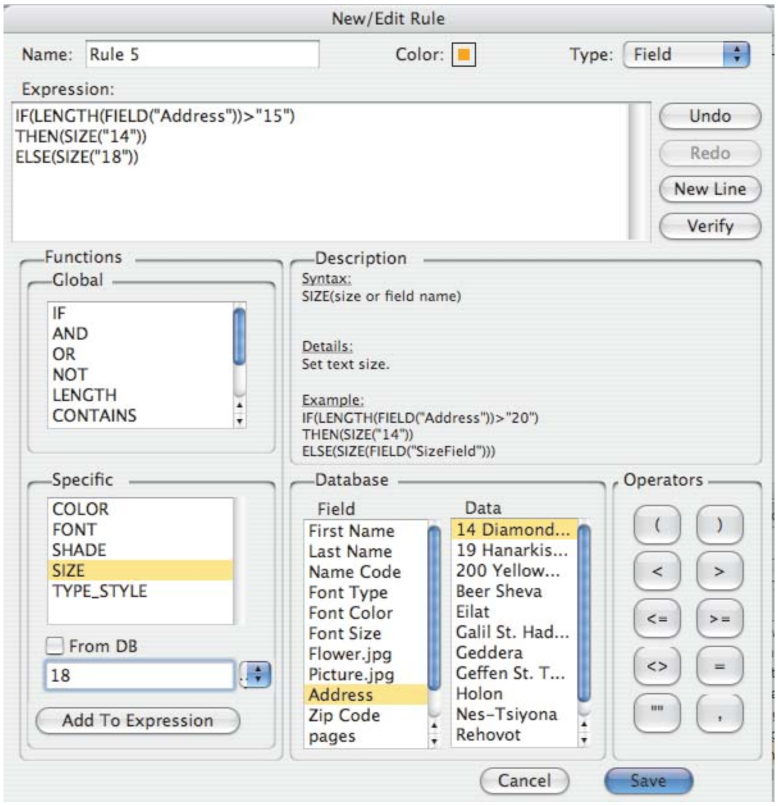
8 You can increase the complexity of the rule (optional):

a To add a new line to the expression, click **New Line**, and edit the new line in the manner described above.

b To use a function other than LENGTH, click on the function name in the Functions pane. The function appears at the cursor location in the Expression field.

9 When you have finished editing your expression, click **Verify**. The function is verified. You are notified of any errors in the expression.

10 When the rule is completed without errors, click **Save**. The rule is saved under the name provided in the Name field.



New/Edit Rule

Name: Rule 5 Color: 🟧 Type: Field

Expression:

```
IF(LENGTH(FIELD("Address"))>"15")
THEN(SIZE("14"))
ELSE(SIZE("18"))
```

Description

Syntax:
SIZE(size or field name)

Details:
Set text size.

Example:

```
IF(LENGTH(FIELD("Address"))>"20")
THEN(SIZE("14"))
ELSE(SIZE(FIELD("SizeField")))
```

Specific: SIZE

☐ From DB

18

Database: Field: Address; Data: 14 Diamond...

Figure 4-10. Completed rule

The example above used a Field Rule. You can perform a similar procedure to create a new Image Rule or Text Rule.

You can also use database attributes in your text rule. See "Using database text attributes with rules" below.

Note You can create complex rules, including nested rules, such as the predefined Picture_Advance1 and Picture_Advance2 rules. See "Predefined Image Rules" on page 82.

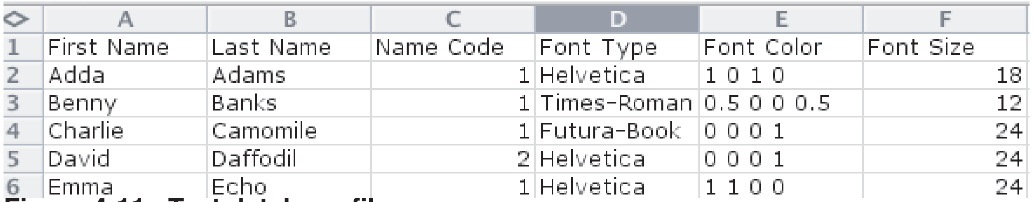
At this point, you can perform one of the following procedures, based on the type of rule that you have created:

- "Assigning rules to text channels" on page 71
- "Assigning rules to image channels" on page 73

Using database text attributes with rules

You can use the text attributes color, font, and size according to the values listed in the DB fields. The DB file will include a separate column that contains text attributes (i.e., a column with text size values). The same DB may also include plain text columns.

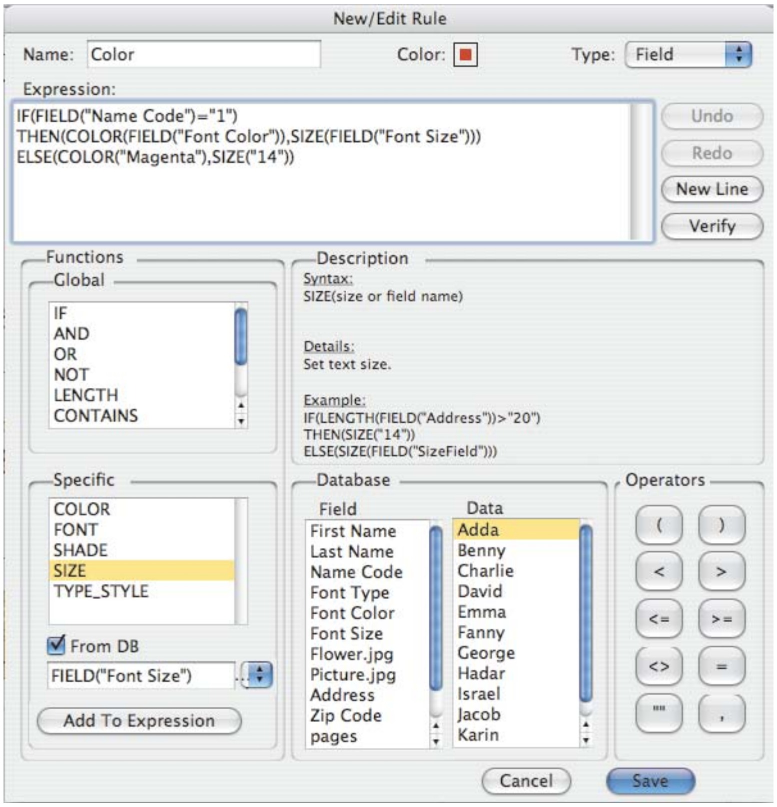
Access this option through either a Text or a Field rule. For example, Figure 4.10 illustrates a database that includes a plain text field for Full Name and text attribute values for Font Type, Font Color, and Font Size.



	A	B	C	D	E	F
1	First Name	Last Name	Name Code	Font Type	Font Color	Font Size
2	Adda	Adams	1	Helvetica	1 0 1 0	18
3	Benny	Banks	1	Times-Roman	0.5 0 0 0.5	12
4	Charlie	Camomile	1	Futura-Book	0 0 0 1	24
5	David	Daffodil	2	Helvetica	0 0 0 1	24
6	Emma	Echo	1	Helvetica	1 1 0 0	24

Figure 4-11. Text database file

Figure 4-12 illustrates a sample rule that includes two fields from the DB file: font color and font size. This rule can be assigned, for example, to the Full Name field.



New/Edit Rule

Name: Color   Color: ■   Type: Field

Expression:

```
IF(FIELD("Name Code")="1")
THEN(COLOR(FIELD("Font Color")),SIZE(FIELD("Font Size")))
ELSE(COLOR("Magenta"),SIZE("14"))
```

Description

Syntax:
SIZE(size or field name)

Details:
Set text size.

Example:
IF(LENGTH(FIELD("Address"))>"20")
THEN(SIZE("14"))
ELSE(SIZE(FIELD("SizeField")))

☑ From DB

FIELD("Font Size")

Figure 4-12. Database rule

To use this feature, perform the following steps:

1 Place the cursor in the desired position in the rule expression.

2 Select an attribute from the **Attributes** pane (in the example: "COLOR").

3 Check the **From DB** check box.

4 Select the desired DB field from the drop-down list that is below the check box.

5 Click the **Add to Expression** button.

The selected field appears as part of the expression text. (See in the example THEN(COLOR(FIELD("FontColor")).

The following values can be used in DB fields to describe text attributes.

Attribute	Values Description	Example
Font	PostScript font name	Helvetica; Times New Roman
Font color	● 1 = 100% ● 0.55 = 50% ● Sep. order CMYK; CMYKOVM1 ● Use space between the separations	● Red: 0 1 1 0 ● 70% Black: 0 0 0 0.7 ● Orange: 0 0 0 0 1 0 0
Font size	Use points units	

The colors used in the Font color values must match the colors that are part of the YTD output file. See “Creating an output file,” step 8 on page 112 for information on selecting colors for an output file.

By default, a YTD output file is a CMYK file in the order C M Y K. For example:

- A color of 100% magenta will be expressed in a CMYK file as 0 1 0 0.
- A color of 100% magenta will be expressed in an IndiChrome job (CMYKOVM1) as 0 1 0 0 0 0 0.
- A color of 100% magenta will be expressed in a 5colors job (where the fifth color was selected to be created) as 0 1 0 0 0 for CMYK5.

Creating a new pages rule

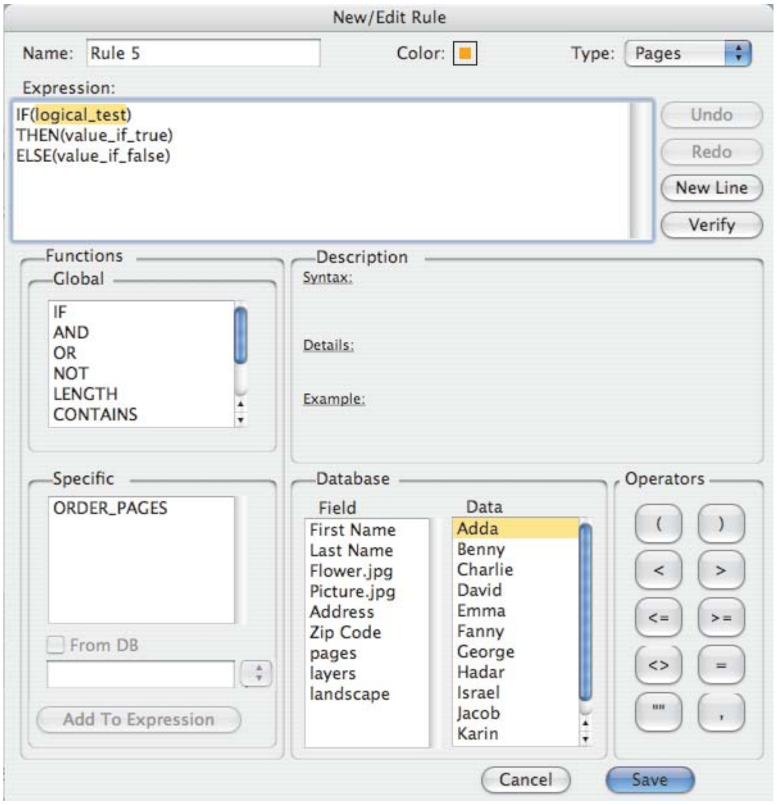
Pages rules are supported only for PPML output files. The PPML workflow is handled by the HP Indigo press 5000 and the HP Indigo Production Flow.

Pages rules are suitable for a multi page QuarkXPress document, where the rule determines which of the document pages appear for each personalization copy. The number of pages and their order varies for the different personalization copies based on the rule definition.

In the following example, the rule is used to determine which of the four QuarkXPress document pages appear for any personalization copy. When the letter is mailed to a ZIP code beginning with "7", it will contain the pages defined by the values in a specific Database column named Pages. When it is mailed to other ZIP codes, the letter will contain only page "1".

To create a new pages rule:

1 Click the **New rule** button, to open the *New/Edit Rule* window.



Name: Rule 5

Type: Pages

Expression:

```
IF(logical_test)
THEN(value_if_true)
ELSE(value_if_false)
```

Figure 4-13. Creating a Pages rule

2 In the Name field, type a name for the rule.

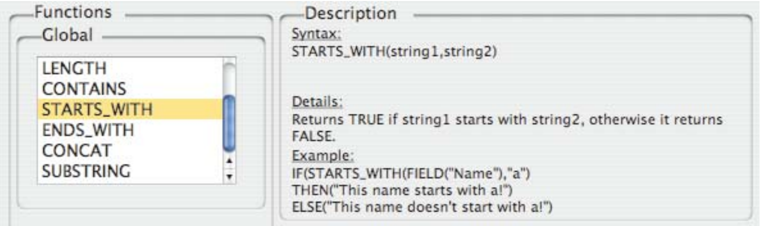
3 You can change the rule color by clicking the **Color** button, and selecting a color from the color palette.

4 In the Type drop-down menu, make sure the rule you use is of the Pages type.

5 Edit the “IF” statement:

a In the Expression field, select the **logical_test** expression.

b In the Functions, Global pane, double-click the desired function (STARTS_WITH in the example). The function appears in the statement.
The function syntax, details, and example are shown on the right of the Functions pane.



Syntax:
STARTS_WITH(string1,string2)

Details:
Returns TRUE if string1 starts with string2, otherwise it returns FALSE.

Example:

```
IF(STARTS_WITH(FIELD("Name"),"a")
THEN("This name starts with a!")
ELSE("This name doesn't start with a!")
```

Figure 4-14. Function syntax and example

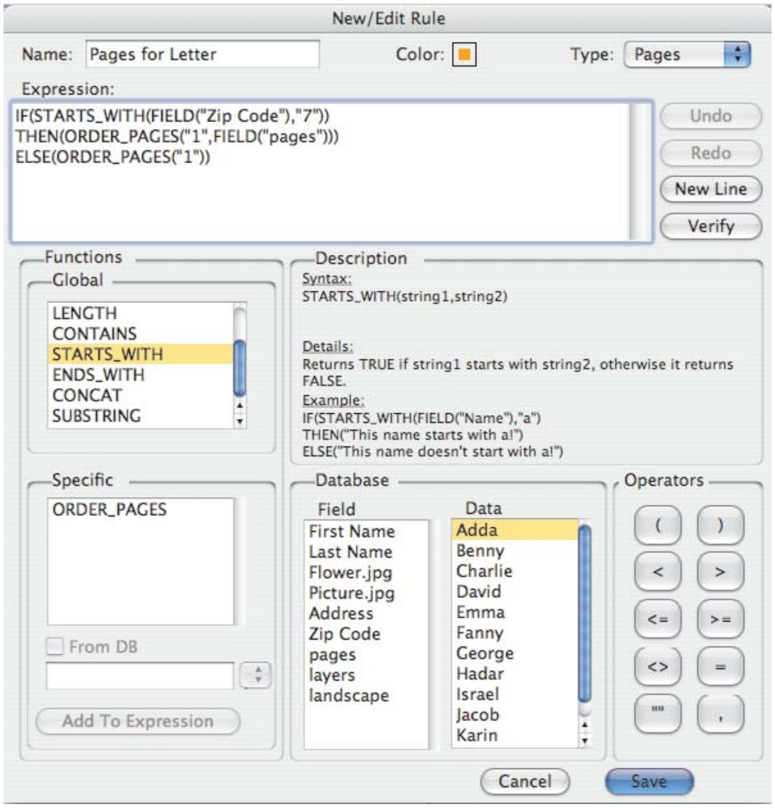
- **c** In the expression, select **string1**.
- **d** In the Database pane, Field list, double-click **Zip Code**. Zip Code replaces string1 in the expression.
- **e** In the expression, select **string2**, and type a value (“7”, including the quotes in this example).

6 Edit the “Then” statement:

- **a** In the Expression field, select the **value-if-true** expression.
- **b** In the Functions pane, Specific list, select **ORDER_PAGES**.
- **c** Check **From DB**.
- **d** In the entry field below **From DB**, type “**1,**” (the number 1 followed by a comma).
- **e** Click on the drop menu arrows, and select **Pages** from the list.
- **f** Click **Add to Expression.**

7 Edit the “Else” statement:

- **a** In the Expression field, select the **value-if-false** expression.
- **b** In the Functions pane, Specific list, select **ORDER_PAGES**.
- **c** Make sure that **From DB** is not checked.
- **d** Click on the drop menu arrows, and select **1** from the list.
- **e** Click **Add to Expression**.



New/Edit Rule

Name: Pages for Letter Color: 🟧 Type: Pages

Expression:

```
IF(STARTS_WITH(FIELD("Zip Code"),"7"))
THEN(ORDER_PAGES("1",FIELD("pages")))
ELSE(ORDER_PAGES("1"))
```

Description

Syntax:
STARTS_WITH(string1,string2)

Details:
Returns TRUE if string1 starts with string2, otherwise it returns FALSE.

Example:
IF(STARTS_WITH(FIELD("Name"),"a"))
THEN("This name starts with a!")
ELSE("This name doesn't start with a!")

Figure 4-15. Completed Pages rule

At this point, you can save the rule, and assign it to you QuarkXPress layout.

Creating a new layers rule

Layers rules are supported only for PPML output files. The PPML workflow is handled by the HP Indigo press 5000 and the HP Indigo Production Flow.

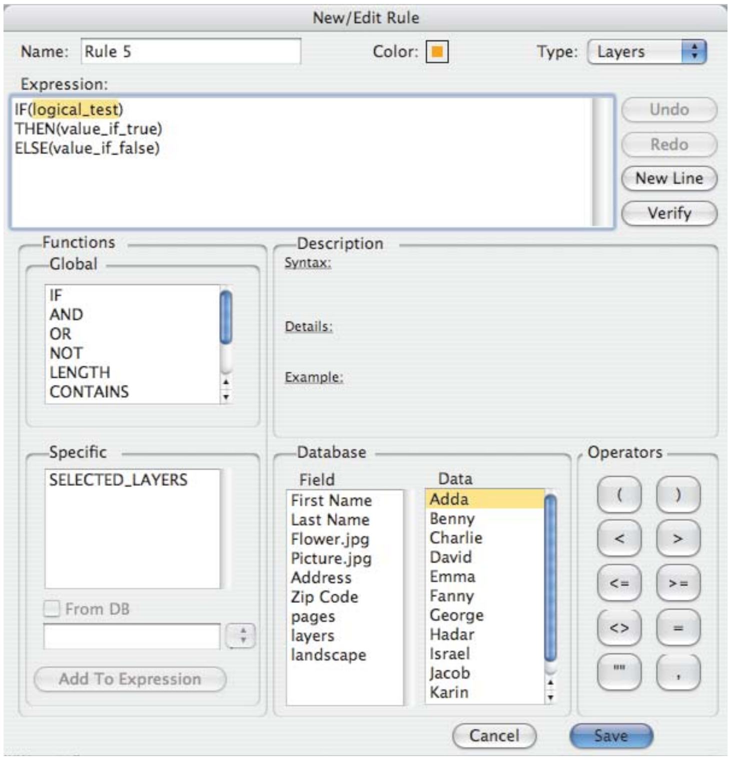
Layers rules can be applied to any QuarkXPress document that contains several layers. The rule is used to define which of the document layers appears for each personalization copy. The number of layers and their identity may vary for the different personalization copies.

In the following example, the rule is used to define which of the 4 QuarkXPress document layers appear for any personalization copy: Each layer is a text box in a different language. The database contains a column that indicates the language: 1= Dutch, 2=French, 3=Italian, 4=Spanish.

A default layer is selected when the address field is USA. The default field and the layer determined by the rule both appear when the address field is not USA.

To create a new layers rule:

1 Click the **New rule** button, to open the *New/Edit Rule* window.



Name: Rule 5

Type: Layers

```
IF(logical_test)
THEN(value_if_true)
ELSE(value_if_false)
```

Figure 4-16. Creating a Layers rule

2 In the Name field, type a name for the rule.

3 You can change the rule color by clicking the **Color** button, and selecting a color from the color palette.

4 In the Type drop-down menu, make sure the rule you use is of the Layers type.

5 Edit the “IF” statement:

 a In the Expression field, select the **logical_test** expression.

 b In the Database pane, Field list, double-click **Address**. Address appears in the expression.

c In the expression, place the cursor before the last bracket. In the Operators pane, click “**<>**”. The operator is added to the expression.

d Type “USA” (including the quotes) at the end of the expression.

6 Edit the “Then” statement:

a In the Expression field, select the **value-if-true** expression.

b In the Functions pane, Specific list, select **SELECTED_LAYERS**.

c Check **From DB**.

d Click on the drop menu arrows, and select **Layers** from the list.

e Un-check **From DB**.

f Click on the drop menu arrows, and select **Default** from the list of available layers.

g Click **Add to Expression.**

7 Edit the “Else” statement:

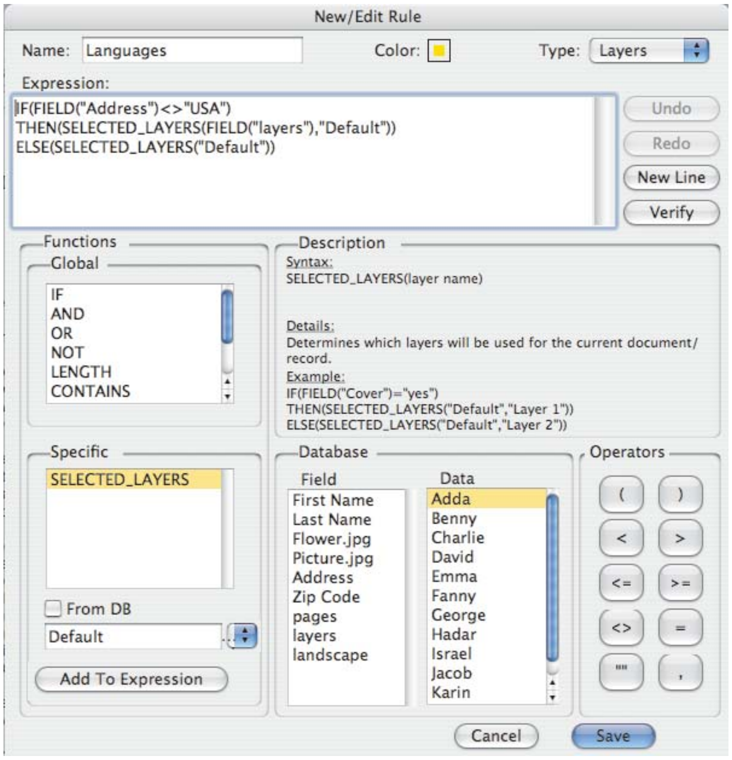
a In the Expression field, select the **value-if-false** expression.

b In the Functions pane, Specific list, select **SELECTED_LAYERS**.

c Make sure that **From DB** is not checked.

d Click on the drop menu arrows, and select **Default** from the list of available layers.

e Click **Add to Expression.**



New/Edit Rule

Name: Languages

Color: 🟨

Type: Layers

Expression:

```
IF(FIELD("Address")<>"USA")
THEN(SELECTED_LAYERS(FIELD("layers"),"Default"))
ELSE(SELECTED_LAYERS("Default"))
```

Description

Syntax:

SELECTED_LAYERS(layer name)

Details:

Determines which layers will be used for the current document/record.

Example:

```
IF(FIELD("Cover")="yes")
THEN(SELECTED_LAYERS("Default","Layer 1"))
ELSE(SELECTED_LAYERS("Default","Layer 2"))
```

Specific: SELECTED_LAYERS

☐ From DB

Default

Figure 4-17. Completed Pages rule

At this point, you can save the rule, and assign it to you QuarkXPress layout.

Assigning rules

Rules can be assigned to fixed text and fields in a text channel, to an entire image channel; or to the entire QuarkXPress document for layers or pages rules.

Assigning rules to text channels

Text channels can contain multiple fields, in addition to fixed data, and multiple rules. You can assign the field rule to text fields, and you can assign the text rule to both text fields and fixed text within a text channel.

Assigning the field rule

To assign a field rule in a text channel:

1 In your QuarkXPress document, select a personalization text channel.

2 In the text channel, select a DB field by doing one of the following:

- Selecting all the characters in the field
- Selecting some of the characters in the field
- Placing the cursor in the field

3 In the Rules palette, select the desired rule from the list of rules.

4 Click the **Assign Field Rule** button. The rule is assigned to the field. An 💡 icon appears on the line of the rule in the Rules list, indicating that the rule has been assigned to the selected field.

Note The **Assign Rule** button is not active if the selected rule type does not match the channel type.

In the QuarkXPress document, the field chevrons (<<>>) are now in the color of the rule, indicating that the rule has been assigned.

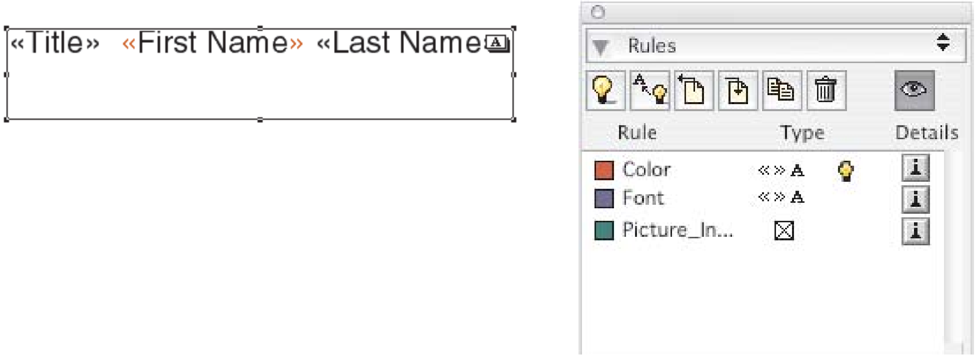


Figure 4-18. Field and text rule assignment indicators

In the Channel palette, you can click the channel's details information button ℹ, to see the name of the rule that is assigned to the channel.

5 You can verify the application of the rule by using the YTD Preview option (see "Previewing the variable job" on page 50).

Note Preview Longest Data option is not supported for the Length rule.

6 To cancel the assignment of the field rule:

a Select the DB field within the personalization text channel.

b In the Rules palette, click the **Assign Field Rule** button. The 💡 icon is removed from the line of the selected rule in the Rules list.
In the QuarkXPress document, the field chevrons (<<>>) revert to their original color.

Assigning the Text Rule

To assign a Text Rule in a text channel:

1 In the QuarkXPress document, select a personalization text channel.

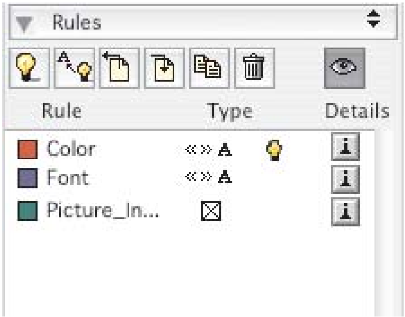
2 In the text channel, select any portion of the text including DB field(s) or fixed data using the QuarkXPress content tool.

3 In the Rules palette, select the desired rule from the list of rules.

4 Click the **Assign Text Rule** button. The bulb icon appears on the line of the rule in the Rules list, indicating that the rule has been assigned to the selected text.

In the QuarkXPress document, the Text rule indicators ({) and (}) appear in the color of the rule, at the beginning and ending of the selected text, indicating that the rule has been assigned to the text.

Note The **Assign Rule** button is not active if the selected rule type does not match the channel type.



Rule	Type	Details
Color	«» A 💡	i
Font	«» A	i
Picture_In...	☒	i

Figure 4-19. Text Rule icon

In the QuarkXPress document, the indicators are now in the color of the rule, indicating that the rule has been assigned.

5 To cancel the assignment of a Text Rule:

a Select either of the text rule indicators (“{“ or “}”).

b Delete the text rule indicator. This automatically deletes the other indicator and cancels the rule assignment.

Assigning rules to image channels

To assign an Image Rule to an image channel:

1 Select an image channel box in your QuarkXPress document.

2 In the Rules palette, select the desired rule from the list of rules.

3 Click the **Assign Image Rule** button. An 💡 icon appears on the line of the selected rule in the Rules list, indicating that the rule is assigned to the channel.

Note The **Assign Rule** button is not active if the selected rule type does not match the channel type.

The Rule indicator appears in the image channel in your QuarkXPress document with the current rule color indicator. This indicates the rule assignment to the channel.

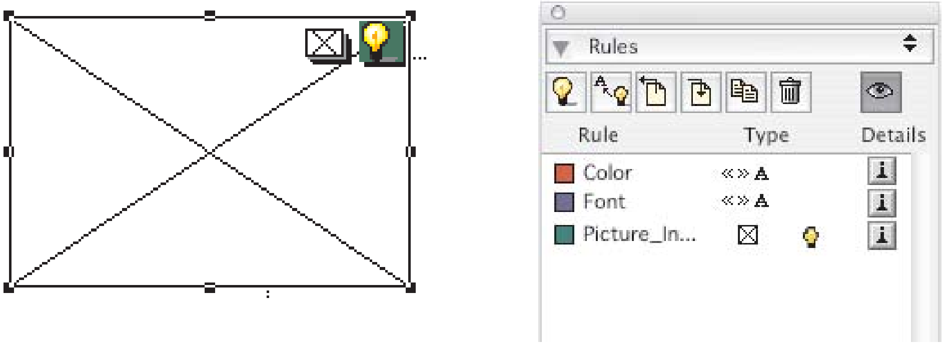


Figure 4-20. Image channel rule assignment indicators

In the Channel palette, you can click on the channel's details information button ℹ, to see the name of the rule that is assigned to the channel.

4 You can verify the application of the rule by using the YTD Preview option (see "Previewing the variable job" on page 50).

5 To cancel the assignment of the Image Rule:

a Select the image channel.

b In the Rules palette, click the **Assign Image Rule** button. The 💡 icon is removed from the line of the selected rule in the Rules list. In the QuarkXPress document, the image channel rule assignment indicator disappears.

Assigning pages or layer rules

Pages rules and layers rules are assigned to the entire QuarkXpress layout. You do not need to create a channel to assign a Pages Rule or a Layers Rule.

To assign a new rule:

1 In the *New/Edit Rule* window, click **Save** after you have finished creating or editing a rule. The following message appears:

Do you want to assign pages rule "Rule 3"?

Cancel
OK

Do you want to assign layers rule "Rule 4"?

Cancel
OK

Figure 4-21. Assign Rule message

2 Click **OK**. The rule is assigned to the entire QuarkXpress layout.

To assign an existing rule:

1 In the Rules palette, select the desired rule from the list.

2 Click the **Assign Pages Rule** button or the **Assign Layers Rule** button.

An 💡 icon appears on the line of the selected rule in the Rules list, indicating that the rule is assigned.

Note Only one Pages rule can be assigned to each QuarkXPress layout.

3 You can verify the application of the rule by using the YTD Preview option (see "Previewing the variable job" on page 50).

4 To cancel the assignment of the Pages Rule or a Layers Rule:

a In the Rules palette, select the rule.

b Click the **Assign Pages Rule** button or the **Assign Layers Rule** button to release it.

The 💡 icon is removed from the line of the selected rule in the Rules list.

Exporting rules

You can export rules from the Rules palette list to the rule folder on your computer. Exporting a rule adds it to the original rule list in the *Mac_HD:Applications:QuarkXPress7:XTensions:Yours Truly Rules* folder. You can copy and move rule files from the rule list to other Macintosh YTD 7 users' rule lists, as you would any other file.

A rule in a QuarkXPress document rule palette can only be used in that specific document. After it is exported, the rule can be used in any other QuarkXPress document.

Note Before exporting a rule, make sure that it has a unique name. For rules based on one of the rules provide with the YTD software, make sure you change the name of the rule before you export it.

To export a rule:

1 Select the rule from the rule list in the Rules palette. You can select more than one rule by using the SHIFT key.

2 Click the **Export rule** button . The rule exports to the *Mac_HD:Applications:QuarkXPress7:XTensions:Yours Truly Rules* folder.

Duplicating rules

You can duplicate rules that you wish to modify.

To duplicate a rule:

1 Select the rule from the rule list in the Rules palette. You can select more than one rule by using the SHIFT key.

2 Click the **Duplicate rule** button. A copy of the rule appears at the bottom of the rule list. The copy is automatically assigned a unique rule color.

You can now edit the duplicated rule, or assign it to a text field or image channel, depending on its type.

Deleting rules

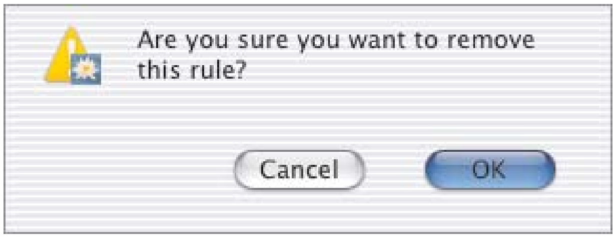
You can delete rules from the rule list in the Rules palette.

Deleting a rule from the rule list deletes it only from the current QuarkXPress document Rules palette list. It does not delete it from the rule folder (*Mac_HD:Applications:QuarkXPress7:XTensions:Yours Truly Rules* folder).

To delete a rule or several rules:

1 Select the rule from the rule list in the Rules palette. You can select more than one rule by using the SHIFT key.

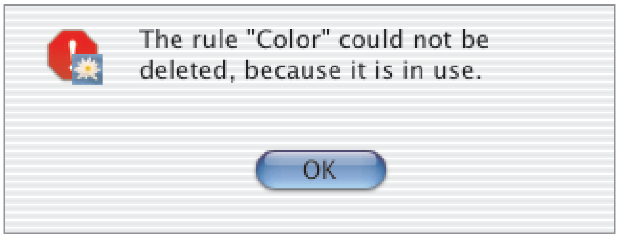
2 Click the **Delete rule** 🗑 button. The Delete rule warning appears.



Are you sure you want to remove this rule?

Figure 4-22. Delete rule warning

3 Click **OK**. All the selected non-assigned rules are deleted from the rule list. If some of the rules that are selected to be deleted are assigned to text field, text channels, image channels, or the entire document; the following message appears and those rules are not deleted.



The rule "Color" could not be deleted, because it is in use.

Figure 4-23. Delete an assigned rule warning

4 Click **OK**.

Syntax elements

Functions

The global functions used in the rule conditional statements are:

Function	Description	Syntax	Example/Explanation

IF	Returns the values following THEN if true.
	Returns the values following ELSE if false.
		IF(logical_test)
		THEN(true value)
		ELSE(false value)
			IF (FIELD (“Gender” = “male”))
			THEN(COLOR(“Cyan”))
			ELSE(COLOR(“Red”))
			When the statement:
			Gender = male
			is true, the text color becomes cyan.
			Otherwise, the text color becomes red.

AND	Returns a true value if all arguments are true. Returns a false value if any of the arguments are false	AND(logical1, logical2) **Note**: You can use more than two arguments in the same rule.	IF (AND(FIELD (“Gender” = “male”, (FIELD(“Age”)<“14”)) THEN(COLOR(“Cyan”)) ELSE(COLOR(“Magenta”)) When both the following statements are true: *Gender is male* *Age is less than 14.* the text color becomes cyan. If either statement is not true, the text color becomes magenta.
OR	Returns a true value if any of the arguments is true. Returns a false value if all the arguments are false	OR(logical1, logical2)	IF (OR(FIELD (“Gender” = “male”, (FIELD(“Age”)<“14”)) THEN(COLOR(“Cyan”)) ELSE(COLOR(“Magenta”)) When either one or both of the following statements are true correct: *Gender is male* *Age is less than 14.* the text color becomes cyan. If both statements are not true, the text color becomes magenta.
NOT	Reverses the logic of its argument.	NOT(logical)	IF (NOT(FIELD(“Gender” = “male”))) THEN(COLOR(“Cyan”)) ELSE(COLOR(“Red”)) When the following statement is false *Gender is male* the text color becomes cyan. If the statement is true the text color becomes red.

Function	Description	Syntax	Example/Explanation
LENGTH	Returns the text length in characters	LENGTH(text or field name)	IF(LENGTH(FIELD(“Name”))>”20”) THEN(SIZE(“12”)) ELSE(SIZE(“36”)) When the following statement is true: *Length of the field “Name” is more than 20 characters* The font size is 12. When the statement is not true, the font size is 36.
CONTAINS	Returns True if string 1 contains string 2, otherwise it returns False	CONTAINS(string1, string2)	IF(CONTAINS(FIELD(“Name”),”a”) THEN(“This name contains a!”) ELSE(”This name does not contain a!”) When the following statement is true *Field “Name” contains the letter “a”* the following appears: *This name contains a!* When the statement is not true, the following appears: *This name does not contain a!*
STARTS WITH	Returns True if string 1 starts with string 2, otherwise it returns False	STARTS_WITH(string1, string2)	IF(STARTS_WITH(FIELD(“Name”),”a”) THEN(“This name starts with a!”) ELSE(”This name does not start with a!”) When the following statement is true: *Field “Name” starts with the letter “a”* the following appears: *This name starts with a!* When the statement is not true, the following appears: *This name does not start with a!*
END WITH	Returns True if string 1 ends with string 2, otherwise it returns False	ENDS_WITH(string1, string2)	IF(ENDS_WITH(FIELD(“Name”),”a”) THEN(“This name ends with a!”) ELSE(”This name does not end with a!”) When the following statement is true: *Field “Name” ends with the letter “a”* the following appears: *This name ends with a!* If the statement is not true, the following appears: *This name does not end with a!*

Function	Description	Syntax	Example/Explanation
CONCAT	Returns the concatenation of the strings which can be fixed text and/or text fields	CONCAT(string1, string2, ...)	IF(FIELD("Month")="May") THEN(CONCAT("Happy birthday ",FIELD("First Name"), "!")) ELSE(CONCAT("Welcome ",FIELD("First Name")," ",FIELD("Last Name"))) If the following statement is true: *Month is May,* the concatenation of the 3 strings appears: *"Happy Birthday " "First name" "!"* If the statement is not true, the concatenation of the 4 strings appears: *"Welcome" "First Name" " " "Last Name"* (A space is considered to be a string)
SUBSTRING	Returns the substring from the start position to the specified length	SUBSTRING(string,start) or SUBSTRING(string,start, len)	IF(LENGTH(FIELD("Address"))>"25") THEN(SUBSTRING(FIELD("Month")," 1","3")) ELSE(FIELD("Month")) When the following statement is true: *Address field contains more than 25 characters* the Month field value is written using the first 3 characters only. The value *1* refers to the starting character, the value *3* refers to the number of characters to be included in the substring. When the statement is not true, the Month field data is written using all the characters

Operators

Operators used in rules are:

Operator	Definition	Description
( )	Open parenthesis Close parenthesis	Encloses a logical test or value
< >	Less than Greater than	Used within a logical test
<= >=	Less than or equal Greater than or equal	Used within a logical test
<>	Not equal to	Used within a logical test
=	Equal	Used within a logical test
““	Quotation marks	Used to define variables or values.
,	Comma	Separates arguments

Rule specific functions

Text/field rule functions

The text attributes that can be part of the rule are limited to the following. Several attributes can be used within the same rule when they are separated by a comma. The attributes can be specific, or linked to a DB field using the DB checkbox:

Attribute	Description
Color	Defines text color
Font	Defines text font
Shade	Defines text color shading
Size	Defines text size in points
Type_Style	Defines type style as Plain, Bold, Italic, and Bold Italic

Image rule functions

Attribute	Description
Image	Defines an image by its file name or linked to a DB field

Pages rule functions

Attribute	Description
ORDER_PAGES	Defines the page order by page names, linked to a DB field, or a combination of both.

Layers rule functions

Attribute	Description
SELECTED_LAYERS	Defines the layer order by layer names, linked to a DB field, or a combination of both.

Predefined rules

Predefined Text/Field Rules

The following predefined Text/Field Rules are provided:

Rule name/Syntax	Explanation

Size
IF(FIELD(“City”)=”Washington”)
THEN(SIZE(“10”))
ELSE(SIZE(“14”))
	Changes the font size of a variable text field based on the contents of the field in the expression.
	If the value in the *City* field is Washington, print using 10 point text, otherwise, use 14 point text.

Rule name/Syntax	Explanation

Font
IF(FIELD(“City”)=”Washington”)
THEN(FONT(“Helvetica”))
ELSE(FONT(“Geneva”),TYPE_STYLE(“Italic”))
	Changes the font of a variable text field based on the contents of the field in the expression.
	If the value in the *City* field is Washington, print using Helvetica font, otherwise, use Geneva font, Italic style.

Length
IF(LENGTH(FIELD(“Address”))>”20”)
THEN(SIZE(“14”))
ELSE(SIZE(“18”))
	Changes the font size of a variable text field based on number of characters in the field.
	If the length of the *Address* field is greater than 20 characters, print using font size 14, otherwise, print using font size 18.

Color
IF(FIELD(“Gender”)=<<male>>A)
THEN(COLOR(“Cyan”))
ELSE(COLOR(“Magenta”),SHADE(“70”))
	Changes the color of a variable text field based on the contents of the field in the expression.
	If the value in the *Gender* field is male, print using Cyan color, otherwise, print using Magenta color, shaded to 70%.

Predefined Field Rule

The following predefined Field Rule is provided:

Rule name/Syntax	Explanation
Free_Text IF(FIELD(“City”)=”Cairo”) THEN(“Come and see the pyramids!”) ELSE(“Welcome and enjoy your visit!”)	**Prints a different free-text string, based on the contents of a variable text field.** If the value in the *City* field is Cairo, print *Come and see the pyramids!*, otherwise, print *Welcome and enjoy your visit!*. You can type in any free text as needed.

Note Field Rules can be used to define text attributes and text contents of variable data only.

Text Rules can be used to define text attributes only for both variable and fixed text.

Predefined Image Rules

The following predefined Image Rules are provided in YTD:

Rule name/Syntax	Explanation

Picture_fix_source
IF(FIELD(“Number”)<“17”)
THEN(IMAGE(“child.jpg”))
ELSE(IMAGE(“adult.jpg”))
	Prints different static image files based on the value of the DB field.
	If the value in the field *Number* is less than 17, insert the image file *child.jpg*, otherwise, insert the image file *adult.jpg*.

Picture_integrated
IF(FIELD(“Gender”)=“male”)
THEN(IMAGE(“Pic1.jpg”))
ELSE(IMAGE(FIELD(“PicName”)))
	Prints a static image file or an image file referenced in the database, based on the value of the DB field.
	If the value in the field *Gender* is male, insert the image file *Pic1.jpg*, otherwise, insert the image file listed in the *PicName* field in the database.

Picture_Advance1
IF(FIELD(“Age”)<“18”)
THEN(IMAGE(“ChildPic.jpg”))
ELSE(IF(FIELD(“Gender”)=”female”)
THEN(IMAGE(FIELD(“WomanPic”)))
ELSE(IMAGE(FIELD(“ManPic”)))
	Prints a static image file if the value in the first DB field is true.
	Prints a referenced image file if the value of the first DB field is false and the second DB field is true.
	Prints a different referenced image file if none of the conditions are true.
	If the value in the field *Age* is less than 18, insert the image *ChildPic.jpg*.
	Otherwise, if the value in the field *Gender* is female, insert the image file listed in the *WomanPic* field in the database.
	If neither of the above is true, insert the image file listed in the *ManPic* field in the database.

Picture_Advance2
IF(FIELD(“Age”)<“6”)
THEN(IMAGE(“Dolly.tif”)
ELSE(IF(FIELD(“Age”)<“22”)
THEN(IMAGE(“Computer.tif”))
ELSE(IMAGE(“Car.tif”)))
	Prints different static image files if the value in the first DB field is true, if the value of the first DB field is false and the second DB field is true, or if none of the conditions are true.
	If the value in the field *Age* is less than 6, insert the image *Dolly.tif*.
	Otherwise, if the value in the field *Age* is less than 22, insert the image *Computer.tif*.
	If neither of the above is true, insert the image *Car.tif*.

Predefined Pages Rule

The following predefined Page Rule is provided in YTD:

Rule name/Syntax	Explanation

IF(FIELD("Customer Code")="A")
THEN(ORDER_PAGES("1",FIELD("pages")))
ELSE(ORDER_PAGES("1"))
	Prints different sets of QuarkXPress document pages for each personalization copy based on rule definitions.
	If the value in the field *Customer Code* is *A*, the personalization copy will contain page 1 followed by the number of pages and their order as defined in the *Pages* DB field.
	If the value in the field *Customer Code* is not *A* only page 1 is printed.

Predefined Layers Rule

The following predefined Layer Rule is provided in YTD:

Rule name/Syntax	Explanation

IF(FIELD("Language")<>"English")
THEN(SELECTED_LAYERS(FIELD("layers"), "Default"))
ELSE(SELECTED_LAYERS("Default"))
	Prints different layers of QuarkXPress design for each personalization copy based on rule definitions.
	If the value in the field *Language* is *English*, the personalization copy will contain layers as defined in the database field named *Layers* and a specific layer named *Default*.
	If the value in the field L*anguage* is not *English* only *Default* layer is printed.

Predefined Concat Rule

The following predefined Concatenation Rule is provided in YTD:

Rule name/Syntax	Explanation

IF(FIELD("Month")="May")
THEN(CONCAT
("Happy birthday ",FIELD("First Name"), "!"))
ELSE(CONCAT("Welcome ",FIELD("First Name")," ",FIELD("Last Name")))
	Prints the concatenation of the strings based on rule definitions.
	If the value in the field *Month* is *May*, the following is printed: *Happy birthday First Name !*
	If the value in the field *Month* is not *May,* the following is printed: *Welcome First Name Last Name*.

Predefined Substring Rule

The following predefined Substring Rule is provided in YTD:

Rule name/Syntax	Explanation

IF(LENGTH(FIELD("Address"))>"25")
THEN(SUBSTRING(FIELD("Month"),"1","3"))
ELSE(FIELD("Month"))
	Prints only substrings of the data in a specific DB field based on rule definitions.
	If the length in the field *Address* is greater than 25 characters, the *Month* field will contain only the first 3 characters of the month name. Otherwise, the complete name of the month is printed.

5 Imposing YTD jobs

This chapter contains the following topics:

- Overview
- PPML and JLYT formats
- Imposing a job using JLYT imposition
- Imposing a job using PPML imposition
- Saving the imposition as a template
- Create an imposition template
- Adding JLYT imposition templates to the press
- Adding spread elements

Overview

Imposition is the process of positioning the pages on spreads to produce final documents, such as booklets or several copies of the same page (such as business cards) on a spread. This is done in order to match finishing requirements, and to optimize paper use. Imposition also consists of positioning and defining cropmarks, bleeds, and gutters.

YTD is supplied with 16 pre-defined imposition templates. These templates can be used as they are, or modified as necessary. There are two types of templates:

- Imposition template files which are saved in the *Mac_Disk:Applications:QuarkXPress 7:XTensions:Yours Truly:Imposition* folder.
- PPML imposition template files which are saved in the *Mac_Disk:Applications:QuarkXPress 7:XTensions:Yours Truly:PPML Imposition* folder.

You can also create new imposition templates that can be used to impose QuarkXPress documents.

Imposition template files can be used to impose jobs on the HP Indigo RIP and/or the HP Indigo press. This process is described in "Adding JLYT imposition templates to the press" on page 115.

PPML imposition templates cannot be used to impose jobs on the HP Indigo RIP and/or the HP Indigo press.

Imposition templates and PPML imposition templates can be generic or non-generic.

- Generic templates are applicable to QuarkXPress documents with any number of pages or page sizes on condition that the pages fit into the spread.
- Non-generic templates are applicable only to QuarkXPress documents that have the same number of pages as were in the QuarkXPress document used to create the template.

Generic template names appear in **bold** while non-generic templates appear in plain text.

All pre-defined imposition templates and PPML imposition templates that are provided with the YTD software are generic templates and new PPML imposition templates.

This chapter describes how to impose a QuarkXPress document, how to use pre-defined imposition templates, and how to create new imposition templates.

PPML and JLYT formats

YTD supports JLYT and PPML output formats.

- JLYT format is the classic YTD output format used for HP Indigo presses. To produce an JLYT output file, use the JLYT imposition.
- PPML (Personalized Print Markup Language) is the new industry-standard printer language for variable data printing which supports a full range of on-demand printing and allows personalized printing to be more flexible, easier to use and more affordable to produce. To produce a PPML output file, use the PPML imposition.

The choice of these formats is set in the YTD preferences window. See "Changing output parameters" on page 147 for details.

Imposing a job using JLYT imposition

Impose a job (referred to as imposition) after a QuarkXPress document is designed and all the personalization channels that you need are added to the job.

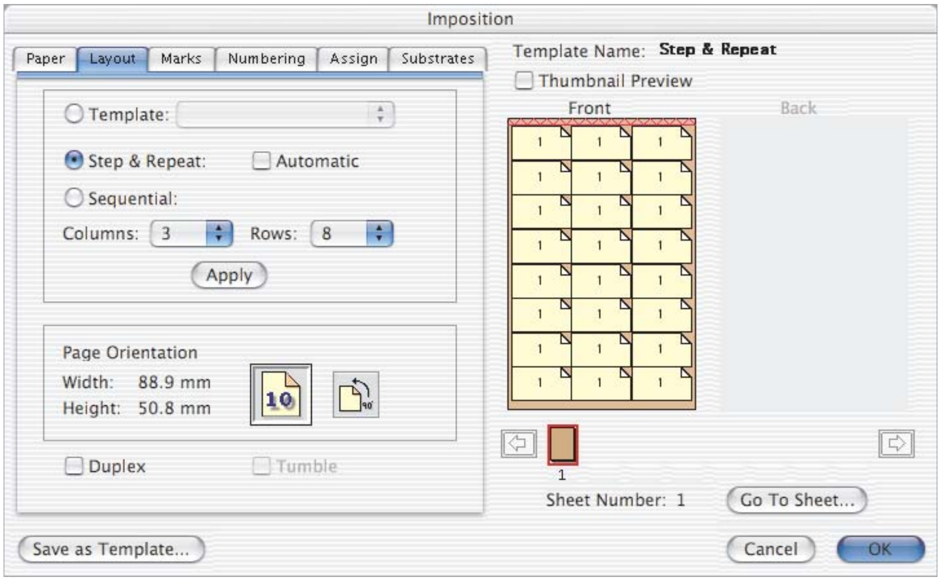
You must impose a job again if pages were added or deleted from the QuarkXPress document.

Imposition is a mandatory step in the YTD workflow, and must be performed before the output file can be created.

Note YTD supports imposition of up to 256 spreads per job.

If not defined by the user, a default imposition of single QuarkXPress page centered on a spread will be used.

Click **Yours Truly** and **Imposition**. The *Imposition* window appears.



Template Name: Step & Repeat

□ Thumbnail Preview

○ Template:

● Step & Repeat: □ Automatic

○ Sequential:

Columns: 3 Rows: 8

Width: 88.9 mm

Height: 50.8 mm

□ Duplex

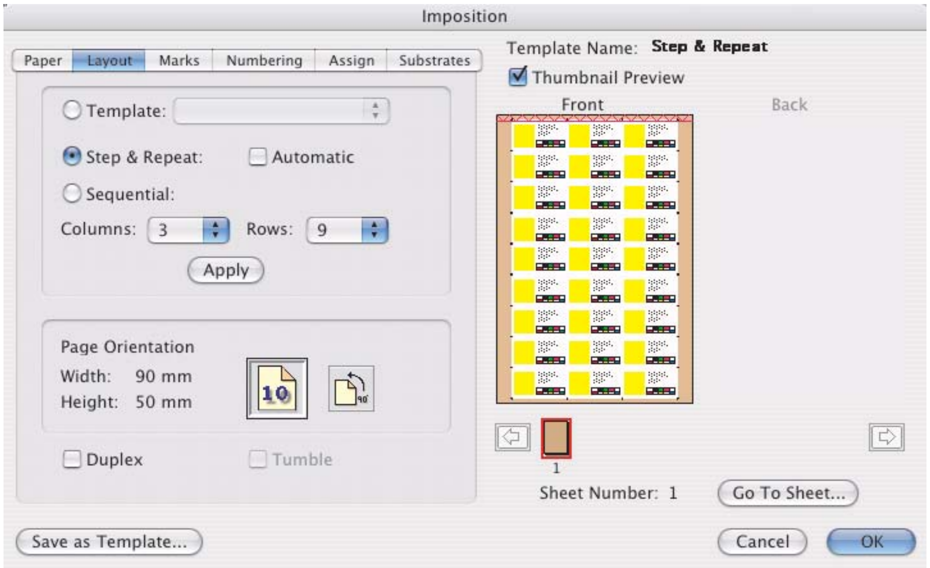
Sheet Number: 1

Figure 5-1. ***Imposition*** **window**

The *Imposition* window contains two main panes:

- The tabs pane on the left contains tabs that give access to different sets of parameters that affect the imposition and printed spreads.

- The preview pane on the right contains a graphical representation of the document pages as they appear on the printed spreads. This graphical representation changes as the parameters in the tabs pane are changed. Select the **Thumbnail Preview** box to enable thumbnails. See Figure 5-2.

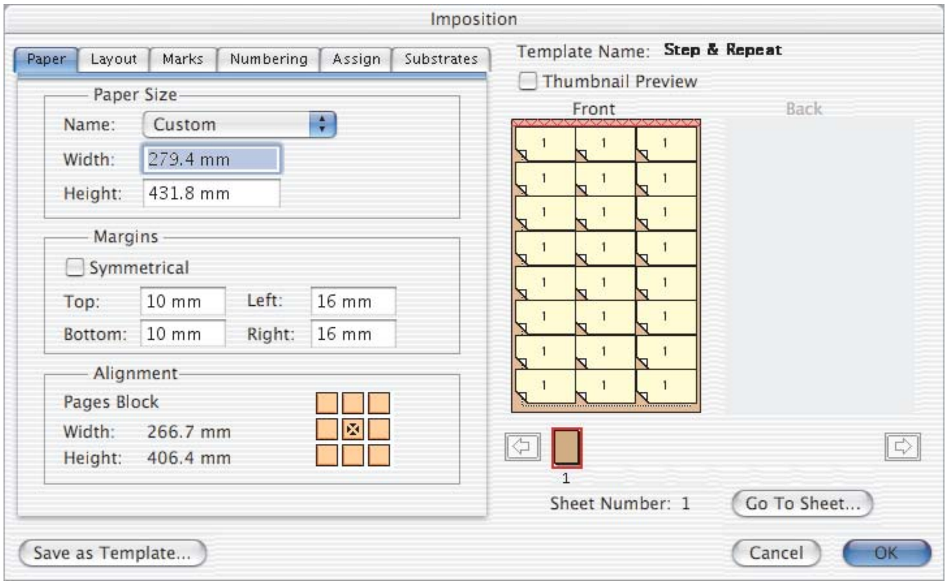


Template Name: Step & Repeat
☑ Thumbnail Preview
○ Template:
● Step & Repeat: □ Automatic
○ Sequential:
Columns: 3 Rows: 9
Page Orientation
Width: 90 mm
Height: 50 mm
□ Duplex □ Tumble
Sheet Number: 1

Figure 5-2. ***Imposition*** **window - thumbnail preview**

Paper tab

1 Click the **Paper** tab.



Template Name: Step & Repeat
□ Thumbnail Preview
Paper Size
Name: Custom
Width: 279.4 mm
Height: 431.8 mm
Margins
□ Symmetrical
Top: 10 mm Left: 16 mm
Bottom: 10 mm Right: 16 mm
Alignment
Pages Block
Width: 266.7 mm
Height: 406.4 mm
Sheet Number: 1

Figure 5-3. Paper tab

2 From the **Paper Size** drop-down menu, select one of the defined sizes. If the size you want does not appear in the list, select **Custom** from the list and define a new paper size (**Width** and **Height**).

3 Verify or change the **Margins** of the paper (spread): **Top/Bottom** and **Left/Right**. The page position cannot go beyond the spread margins and borders.

If you want symmetry of top/bottom and left/right, select the **Symmetrical** check box.

4 In the Alignment pane, you can align the block of pages to nine different positions within the margins: top/middle/bottom and left/center/right.

Use the alignment positioning keys (shown below) to align the pages.

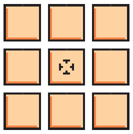


Figure 5-4. Alignment keys

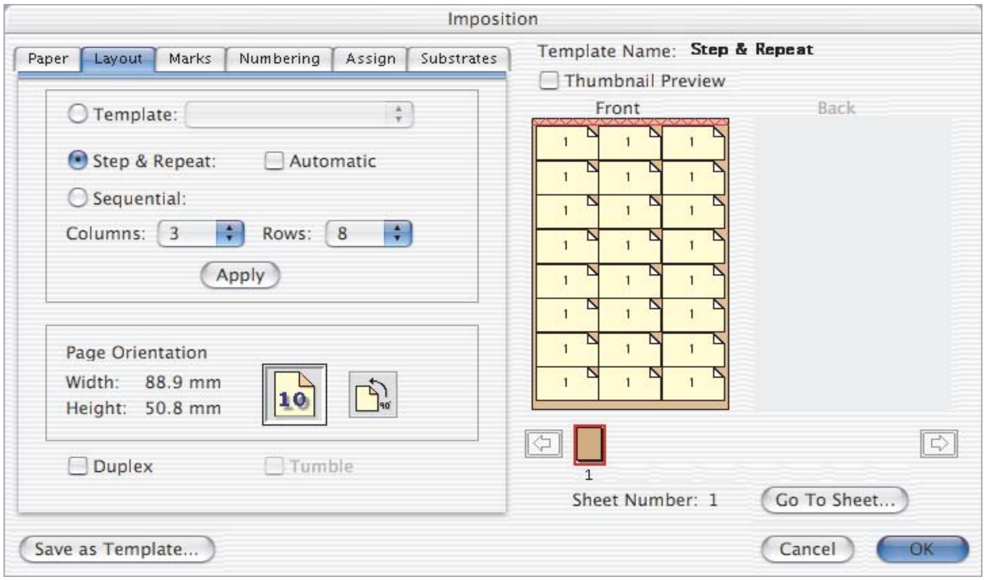
Note

The Width and Height of the Pages Block (as displayed in the Alignment pane) are based on the sum of the pages' widths and heights, and on the sizes of the crop marks, bleed, and gutters.

The default values of the **Paper Size** and **Margins** fields can be modified. Additionally, a new paper can be added to the list. See "Changing paper parameters" on page 140.

Layout tab

1 Click the **Layout** tab.



Imposition

Template Name: Step & Repeat

□ Thumbnail Preview

○ Template:

● Step & Repeat: □ Automatic

○ Sequential:

Columns: 3 Rows: 8

Page Orientation

Width: 88.9 mm

Height: 50.8 mm

□ Duplex □ Tumble

Sheet Number: 1

Figure 5-5. Layout tab

2 Select the option for the type of imposition you want:

- **Template**:

  When you select this radio button, its drop-down menu becomes available. From the drop-down menu, select the YTD-supplied or user-defined template that meets your needs. YTD supplies 16 predefined generic templates. Each template has its own set of values for the various fields. Generic template names are **bold** while non-generic templates are plain.

- **Step & Repeat**:

  Selecting this radio button causes the first page to be repeated many times on the first spread, the second page to be repeated many times on the second spread, and so on.

  Select the relevant values in the **Columns** and **Rows** drop-down menus. Alternatively, check the **Automatic** check box to repeat the page as many times as needed to fill the spread completely with pages within the margins.

- **Sequential** (for QuarkXPress multi-page documents only):

  When you select this radio button, the pages are printed sequentially, starting from the first spread onwards. The order of the pages is dependent on the parameter values set in the Assign tab. Select the relevant values in the **Columns** and **Rows** drop-down menus.

3 Click **Apply** to apply your selections to the job.

4 You may want to change the following parameters, as follows:

a If you want to rotate the pages 90^{o} counter clockwise, click the **Rotation** icon . This may be done repeatedly.

b Check the **Duplex** check box to print duplex.

c Select or clear the **Tumble** check box to control the duplex printing direction. During duplex printing, the press inverts the sheet (leading edge to trailing edge) after printing the first side and before printing the second (duplex) side. The duplex side is printed upside down relative to the first side.

To force the duplex side to be printed in the same direction as the first side, YTD automatically rotates the duplex side 180°. This is referred to as tumble.

If you want to change this, select or clear the **Tumble** check box.

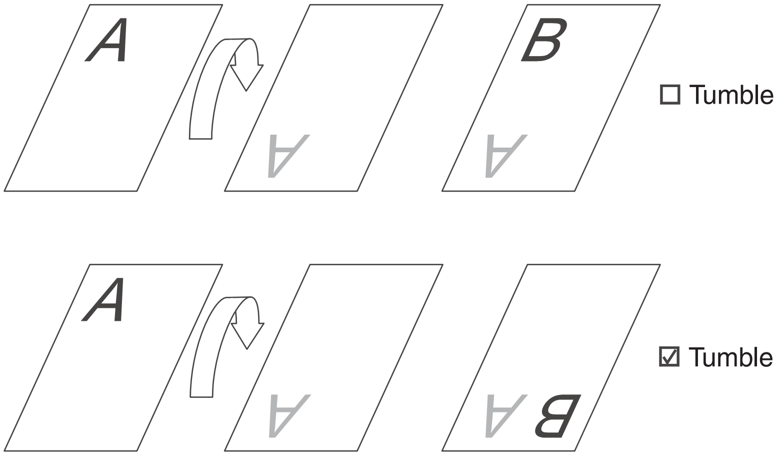


Figure 5-6. Effects of the Tumble check box

Predefined fields

The following fields are predefined by the template you select:

- In the **Paper** tab, the Margins and Alignment fields.
- In the **Layout** tab, the **Duplex** check box (except for those templates ending with _sim).
- In the **Marks** tab, the Crop Marks, Bleed, and Gutters fields.

- The **Assign** tab fields are available when personalization channels are included in the job or when the layout type is Sequential.
- In the **Substrate** tab, substrate types are predefined.
- In the **Layout** tab, the following fields are predefined.

Template name	Page orientation	Tumble
1_up	as is	☑
2_up_cnc	rotated ccw	
2_up_cnc_nr	as is	
2_up_cnc_nr_sim	as is	
2_up_pb	rotated ccw	
2_up_pb2	as is	☑
2_up_ss	rotated ccw	
2_up_ss2	as is	☑
2_up_step_sim	rotated ccw	
3_up	rotated ccw	
4_up_cnc	as is	☑
4_up_pb	as is	☑
4_up_ss	as is	☑
4_up_step_sim	as is	☑
8_up_pb	rotated ccw	
8_up_ss	rotated ccw	

Abbreviations:

- rotated ccw - rotated counterclockwise
- cnc - cut and collate
- nr - no rotation

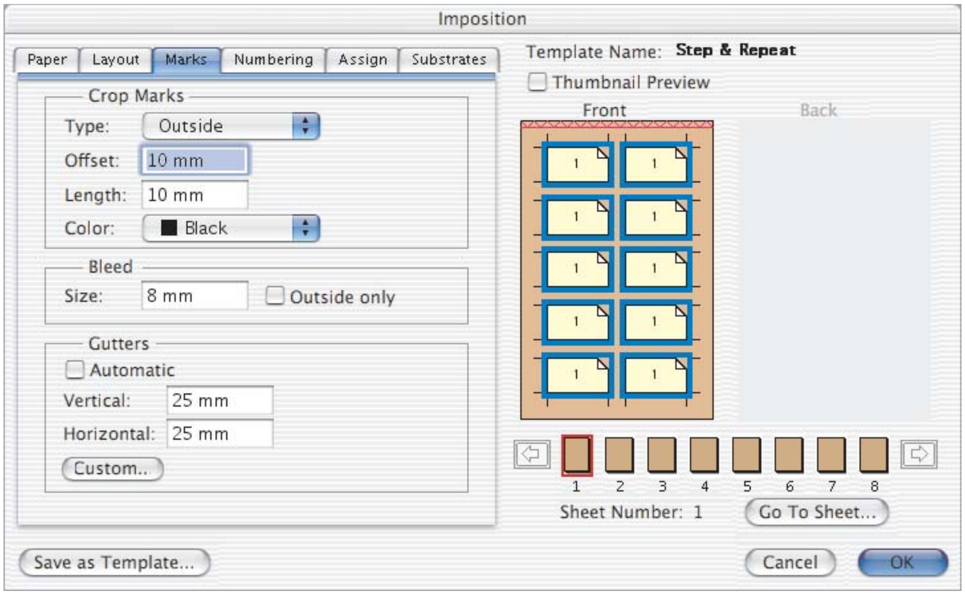
- pb - perfect bound
- sim - simplex
- ss - saddle stitch
- step - step & repeat

Note The Step & Repeat and Sequential layout types do not predefine any fields.

You can change the values of predefined fields, per document, after the template is selected.

Marks tab

1 Click the **Marks** tab.



Imposition

Template Name: Step & Repeat

☐ Thumbnail Preview

Crop Marks

Type: Outside

Offset: 10 mm

Length: 10 mm

Color: ■ Black

Bleed

Size: 8 mm ☐ Outside only

Gutters

☐ Automatic

Vertical: 25 mm

Horizontal: 25 mm

Sheet Number: 1

Figure 5-7. Marks tab

2 Select the appropriate settings in the Crop Marks pane:

a In the **Type** drop-down menu, select one of the following:

- **None**, for no crop marks.
- **Outside**, for crop marks on the outside perimeter of the block of pages.
- **Every Page**, for eight crop marks on each page.

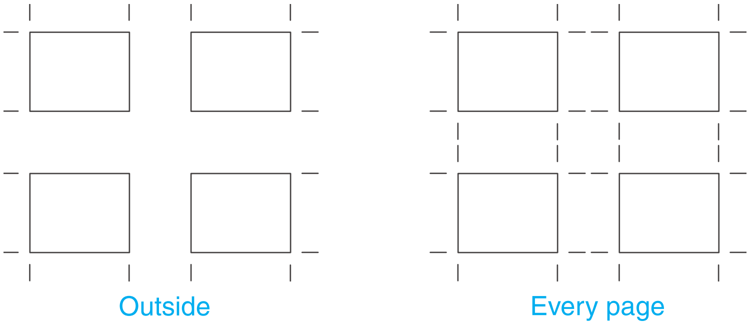


Figure 5-8. Types of crop marks

b In the **Offset** field, type the distance between the end of the page and the beginning of the crop mark.

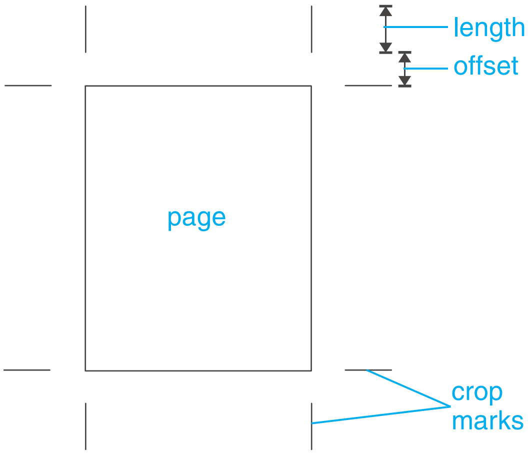


Figure 5-9. Example of offset

c In the **Length** field, type the length of the crop mark.

d In the **Color** drop-down menu, select the color of the crop marks. You can select **Registration** to have the crop marks printed in all the separations of the job.

3 Select the appropriate settings in the Bleed pane:

a In the **Size** field, type the size of the bleed area.

b If you want the bleed area to be outside the block of pages only, select the **Outside only** check box. If this check box is not selected, the bleed will be in the gutters too.

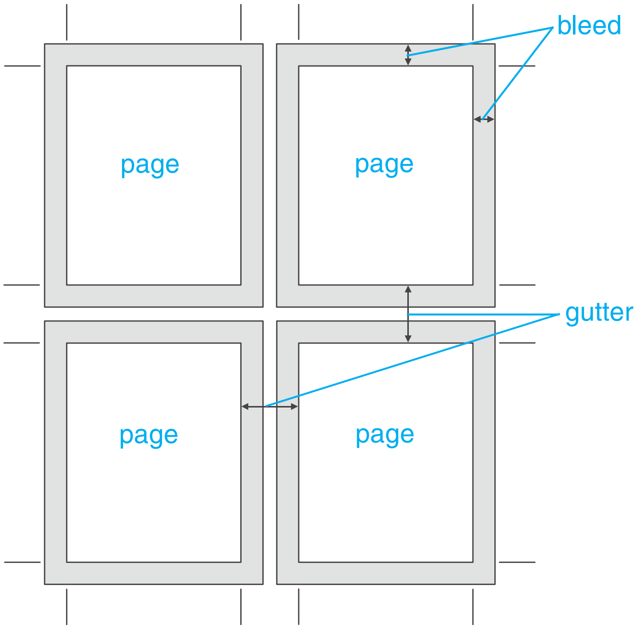


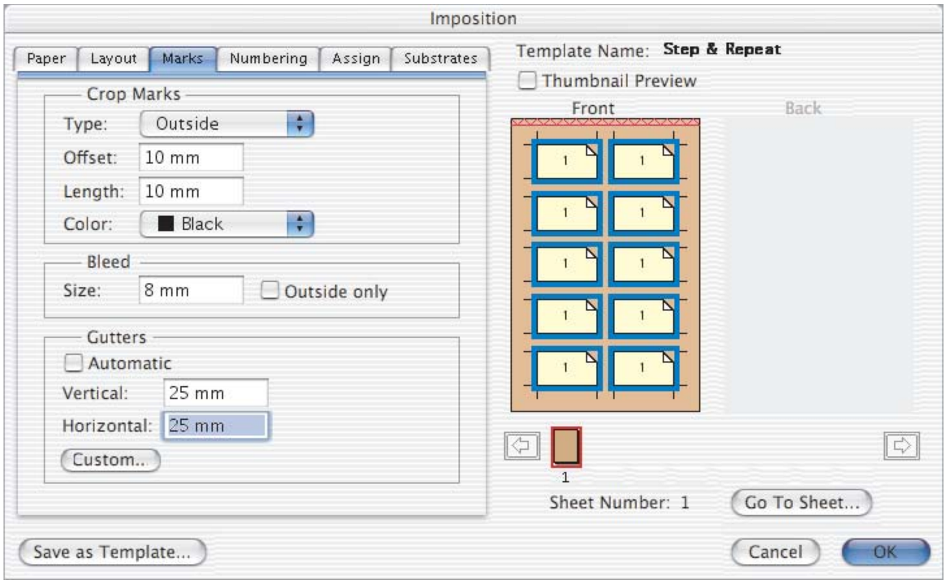
Figure 5-10. Bleed and gutters

Note The default values of the Crop Marks and Bleed fields can be changed. See "Changing marks parameters" on page 143.

4 Select the appropriate settings in the Gutters pane:

a In the **Vertical** and **Horizontal** fields, type the size of the space (gutter) between the columns (vertical) and rows (horizontal) of the pages.

If the vertical or horizontal gutter size measures less than the bleed size, there will be no bleed in the column's or row's gutter:



Template Name: Step & Repeat

□ Thumbnail Preview

Crop Marks
Type: Outside
Offset: 10 mm
Length: 10 mm
Color: ■ Black

Bleed
Size: 8 mm □ Outside only

Gutters
□ Automatic
Vertical: 25 mm
Horizontal: 25 mm

Sheet Number: 1

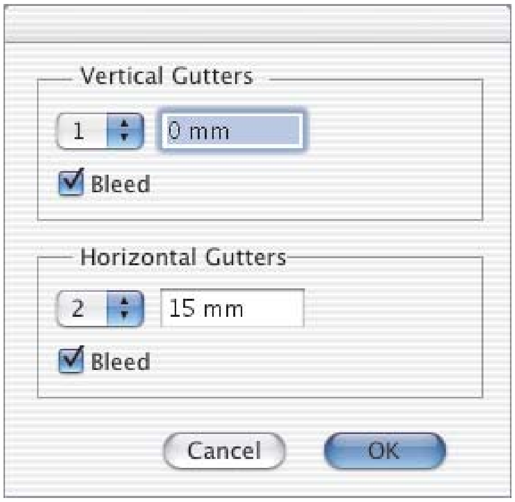
Figure 5-11. No bleed in gutters

b Select the **Automatic** check box to automatically set the gutter size to twice the sum of the crop marks offset plus length or to twice the bleed size, whichever is greater. If the crop mark type is None or Outside, its offset and length do not affect the gutter calculations. Similarly, if the bleed is Outside only, its size does not affect the gutter calculations.

Examples (numbers are in mm):

Offset	Crop mark length	Bleed size	Automatic gutter size
5	2	6	14
5	2	11	22
5	2	11 Outside	14
5	2 Outside	6	12
5	2 Outside	11 Outside	0

c To give different values to different vertical and horizontal gutters and bleed, click **Custom**. The *Vertical/Horizontal Gutters* window appears.



Vertical Gutters

1 | 0 mm

☑ Bleed

Horizontal Gutters

2 | 15 mm

☑ Bleed

Figure 5-12. Custom gutters and bleed

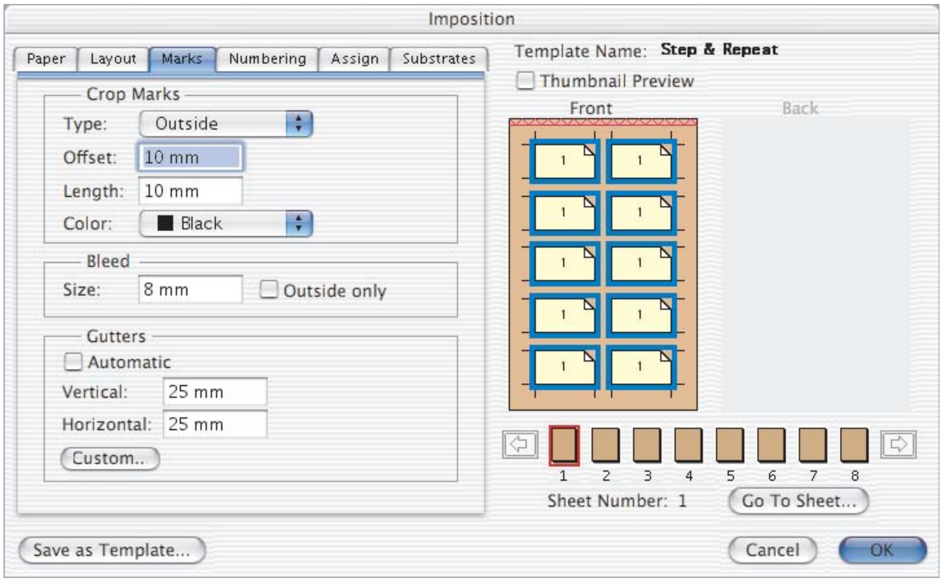
In the Vertical Gutters and Horizontal Gutters panes, do the following:

i In the number drop-down menus, select the gutter number, left to right and top to bottom.

ii Type the size of the gutter in the field next to the gutter number.

iii To clear the bleed in the selected gutter, clear the **Bleed** check box.

If the gutters or bleed are customized, the word `*custom` appears in the Gutters pane and red arrows appear in the preview pane.



Imposition

Crop Marks

Type: Outside

Offset: 10 mm

Length: 10 mm

Color: ■ Black

Bleed

Size: 8 mm □ Outside only

Gutters

□ Automatic

Vertical: 25 mm

Horizontal: 25 mm

Template Name: Step & Repeat

□ Thumbnail Preview

Sheet Number: 1

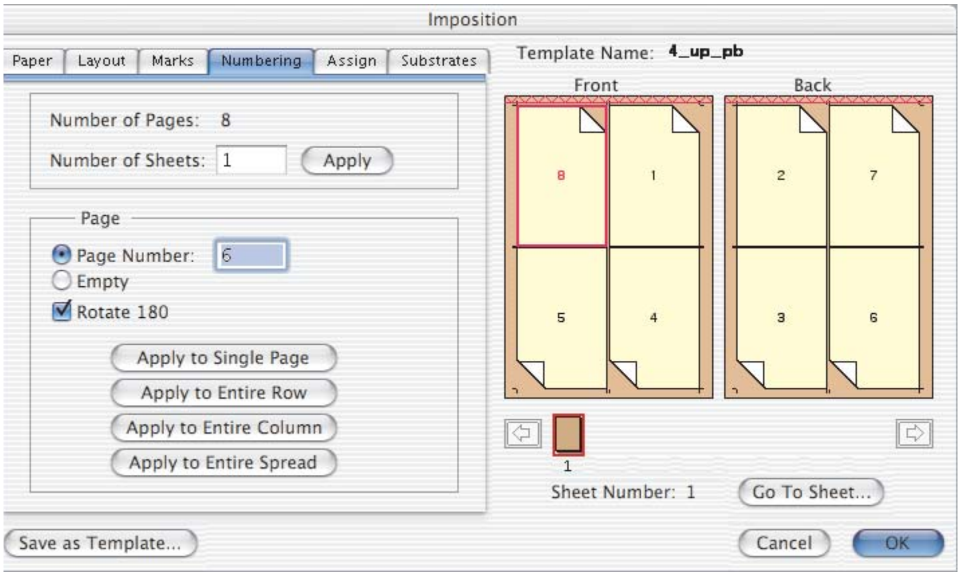
Figure 5-13. Customized bleed and gutters

Numbering tab

In the **Numbering** tab you can replace a page by a specific page on the spread, and define the number of sheets in the imposition.

Note This tab is unavailable if you selected Step & Repeat and Automatic, or Sequential, in the **Layout** tab (see "Layout tab" on page 89). However, if you selected Step & Repeat and Automatic, or Sequential, and you want to edit the fields in this tab, you can save the imposition values as a user-defined template (see “Save as template” on page 102). After you do this, the **Numbering** tab fields become available.

1 Click the **Numbering** tab.



Template Name: 4_up_pb

Number of Pages: 8

Number of Sheets: 1

Page

● Page Number: 6

○ Empty

☑ Rotate 180

Sheet Number: 1

Figure 5-14. Numbering tab

2 The **Number of Pages** field displays the number of QuarkXPress pages in the job.

In the **Number of Sheets** field, you can change the number of sheets of paper upon which the QuarkXPress pages will be distributed. To do this, type the number of sheets in this field and then click **Apply**.

3 To change one or more of the pages, do the following:

a In the preview pane, click the page that you want to change.

b Select the type of change:

- To replace the page with a specific page, click the **Page Number** radio button and type the page number.
- To replace the page with an empty page, click the **Empty** radio button.
- To rotate the page 180^{o}, select the **Rotate 180** check box.

c Click the relevant action button:

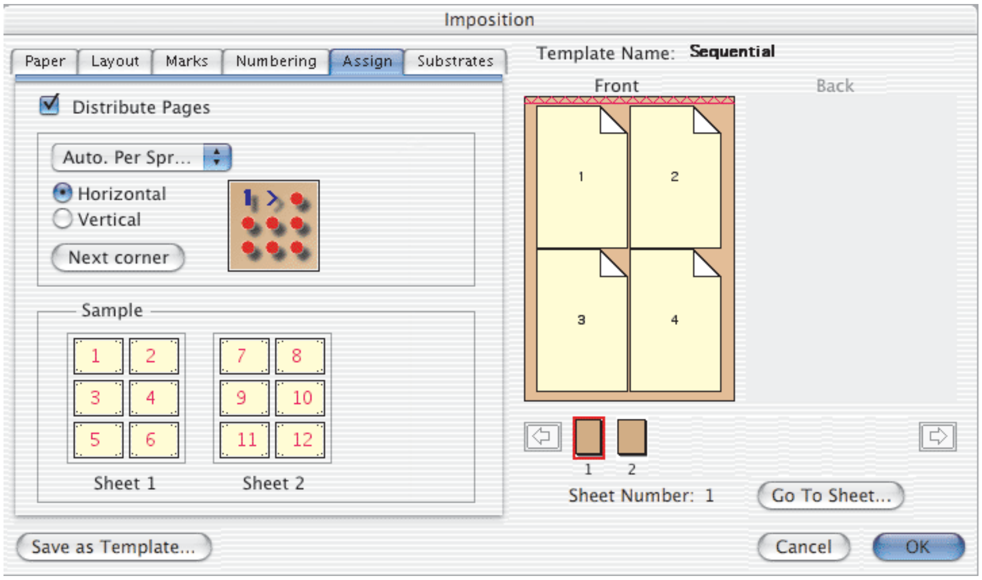
- Apply to Single Page
- Apply to Entire Row
- Apply to Entire Column
- Apply to Entire Spread

Assign tab

The **Assign** tab fields are available when personalization channels are included in the job or when the layout type is Sequential.

Personalization channels included in the job

1 Click the **Assign** tab.



Template Name: Sequential

☑ Distribute Pages

Auto. Per Spr...

● Horizontal

○ Vertical

Sheet Number: 1

Figure 5-15. Assign tab - step & repeat

The Assign tab allows you to define the print order of the channel cycles. The Sample pane on the Assign page shows an example of the results that the assigning has on a typical job. See figure 5-15.

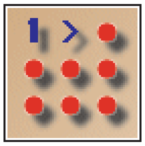
2 Select the assign method from the **Auto. Per...** drop-down menu:

- **Auto. Per Spread** assigns the channel cycles per spread. The same assignment order is applied to each spread.
- **Auto. Per Job** assigns the channel cycles per job. The assignment is applied to the entire job, across all the spreads.
- **Auto. Per Line** assigns the channel cycles per line across the entire job.

3 Click **Next corner** to position the first channel cycle in a page that is a different corner of the spread.

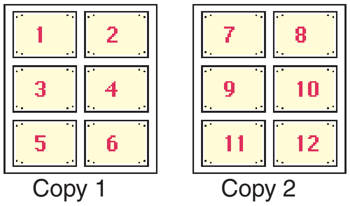
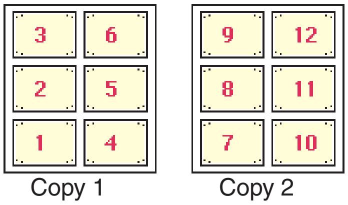
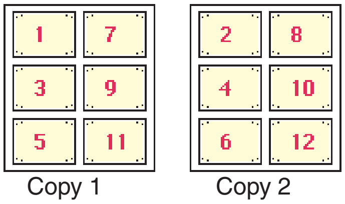
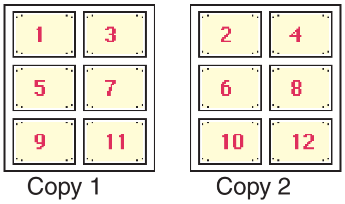
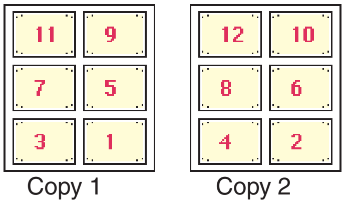
4 Choose the assign direction by selecting the **Horizontal** or **Vertical** radio button.

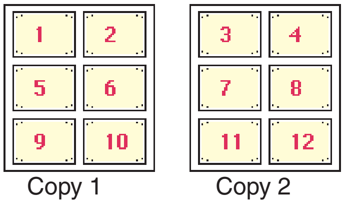
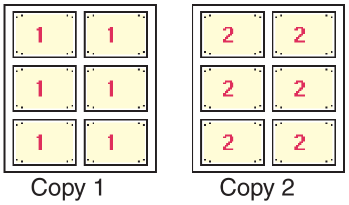
Note The positioning icon shows the position of the first channel cycle and the direction of the subsequent cycles.



5 To assign the first channel cycle to all the pages on the first spread, the second channel cycle to all the pages on the second spread, the third channel cycle to all the pages on the third spread, and so on, clear the **Distribute Data** check box.

Below are examples of Assign results for personalization jobs.

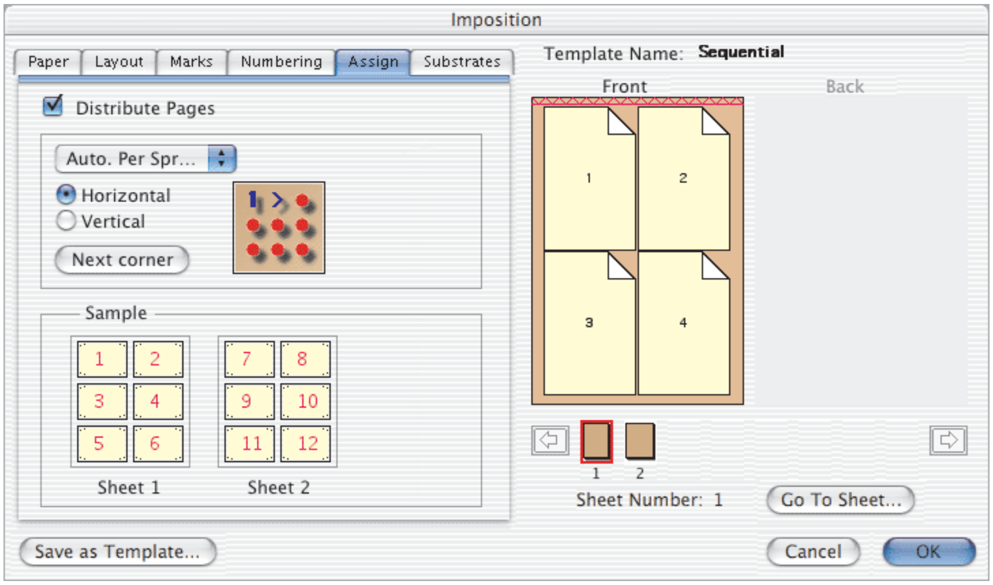
Assign method	Corner position	Direction	Channel cycle assignment
per spread	top left	horizontal	 1 2 3 4 5 6 Copy 1 7 8 9 10 11 12 Copy 2
per spread	bottom left	vertical	 3 6 2 5 1 4 Copy 1 9 12 8 11 7 10 Copy 2
per job	top left	vertical	 1 7 3 9 5 11 Copy 1 2 8 4 10 6 12 Copy 2
per job	top left	horizontal	 1 3 5 7 9 11 Copy 1 2 4 6 8 10 12 Copy 2
per job	bottom right	horizontal	 11 9 7 5 3 1 Copy 1 12 10 8 6 4 2 Copy 2

Assign method	Corner position	Direction	Channel cycle assignment
per line	top left	horizontal	 1 2 5 6 9 10 Copy 1 3 4 7 8 11 12 Copy 2
Distribute Data check box is cleared			 1 1 1 1 1 1 Copy 1 2 2 2 2 2 2 Copy 2

6 Proceed to "Concluding the JLYT imposition" on page 102.

Sequential layout type

1 Click the **Assign** tab.



Imposition

Paper | Layout | Marks | Numbering | Assign | Substrates

☑ Distribute Pages

Auto. Per Spr...

● Horizontal
○ Vertical

Next corner

Sample

Sheet 1

1	2
3	4
5	6

Sheet 2

7	8
9	10
11	12

Template Name: **Sequential**

Front

Back

Sheet Number: 1

Go To Sheet...

Save as Template...

Cancel

OK

Figure 5-16. Assign tab - sequential

The Assign page allows you to reposition (assign) the QuarkXPress pages on the sheets. The sample pane shows an example of the results that the assigning has on a typical job. The preview pane shows the actual results on your job.

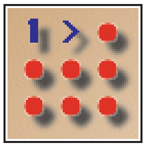
2 Select the assign method from the **Auto. Per...** drop-down menu:

- **Auto. Per Spread** assigns the pages per sheet. The same page assignment order is applied to each sheet.
- **Auto. Per Job** assigns the pages per job. The page assignment is applied to the entire job, across all the sheets.

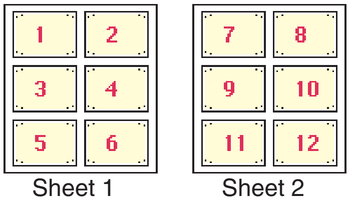
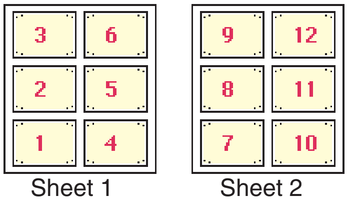
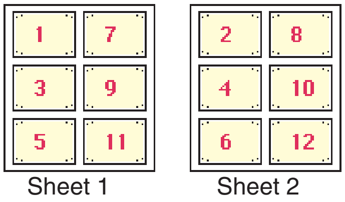
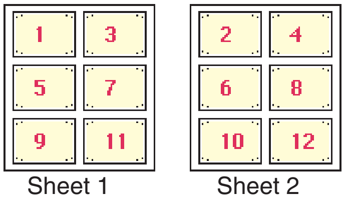
3 Click **Next corner** to position the first page in a different corner of the sheet.

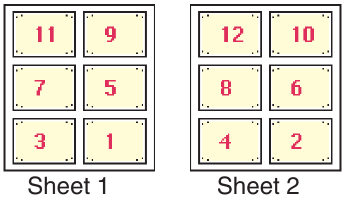
4 Choose the assign direction by selecting the **Horizontal** or **Vertical** radio button.

Note The positioning icon shows the position of the first page and the direction of the subsequent pages.



Below are examples of Assign results.

Assign method	Corner position	Direction	Page assignment
per spread	top left	horizontal	 Sheet 1: 1 2 / 3 4 / 5 6 Sheet 2: 7 8 / 9 10 / 11 12
per spread	bottom left	vertical	 Sheet 1: 3 6 / 2 5 / 1 4 Sheet 2: 9 12 / 8 11 / 7 10
per job	top left	vertical	 Sheet 1: 1 7 / 3 9 / 5 11 Sheet 2: 2 8 / 4 10 / 6 12
per job	top left	horizontal	 Sheet 1: 1 3 / 5 7 / 9 11 Sheet 2: 2 4 / 6 8 / 10 12

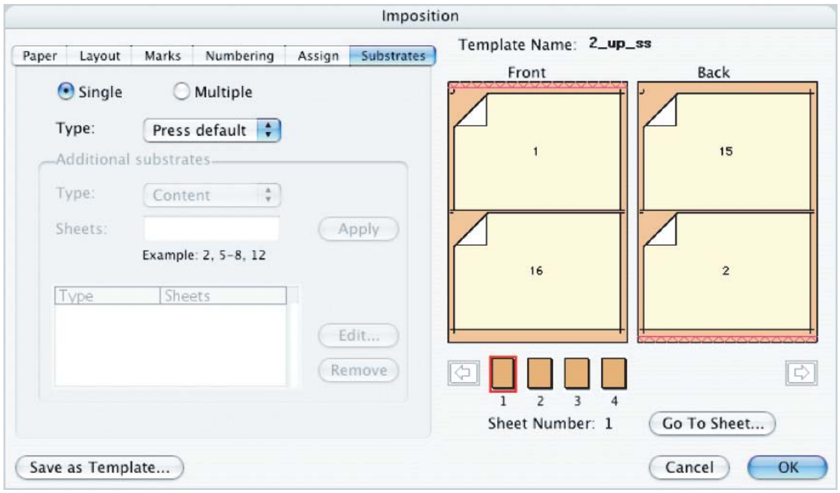
Assign method	Corner position	Direction	Page assignment
per job	bottom right	horizontal	

5 Proceed to "Concluding the JLYT imposition" on page 102.

Substrates tab

Only the HP Indigo press 5000 fully supports the Substrate definition. Other HP Indigo presses will ignore Substrate definitions sent to those presses.

The default Substrate definition is a single substrate, applied to all the sheets, named Press Default, as shown in Figure 5-17. To change the single substrate to another select a name from the drop-down list. The same reserved words on the drop-down list appear also at the HP Indigo Press 5000 substrate menu.



Imposition

Template Name: 2_up_ss

● Single ○ Multiple

Type: Press default

Example: 2, 5-8, 12

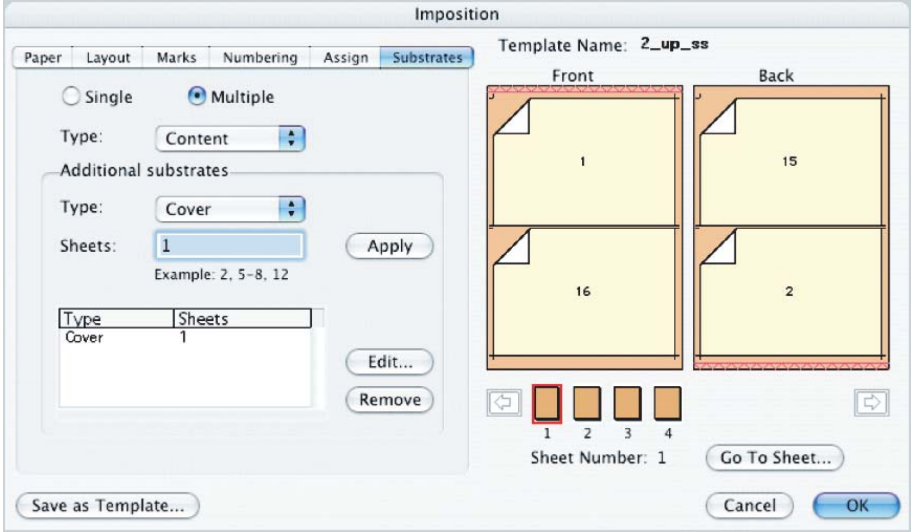
Sheet Number: 1

Figure 5-17. *Imposition* window for default single substrate

1 Make sure the **Single** radio button is selected.

2 From the predefined list of substrates in the **Type** field, select the substrate type.

It is possible to define more than one substrate to a multi-sheet job at the *Imposition* window as shown in figure 5-18. For example, you could define “Cover” for sheet number one and “Content” for all other sheets in the job.



Imposition

Template Name: 2_up_ss

○ Single ● Multiple

Type: Content

Additional substrates

Type: Cover

Sheets: 1

Example: 2, 5-8, 12

Type	Sheets
Cover	1

Sheet Number: 1

Figure 5-18. ***Imposition*** **window for multiple substrate**

3 Click the **Multiple** radio button.The *Additional substrates* pane is enabled.

4 In the upper **Type** drop-down menu, select a general substrate name that will be used for all the sheets in the job, except those which will be defined in the *Additional substrates* pane.

In addition to the original reserved words in the list, you can add additional substrate names in the *Preferences* window in the **Paper** tab.

These additional YTD substrate names are accepted as intended substrates at the press. At the press, the intended names are then mapped to the actual substates names, which are names from the press substrate list.

Using intended substate names in YTD which are identical to actual substate names used on the specific HP Indigo press, ensures a more efficient workflow.

See “Changing paper parameters” on page 136 for more information about setting preferences.

5 Edit or delete the definitions in the display table.

Concluding the JLYT imposition

1 Click **OK**.

2 If you want to add spread elements, proceed to "Adding spread elements" on page 118.

Otherwise, proceed to "Creating the output file" on page 119.

Imposing a job using PPML imposition

PPML imposition expands the range of functions compared to JLYT imposition. In addition to the JLYT imposition functions described above, PPML imposition provides:

- Unlimited Step & Repeat
- Composition of Step & Repeat and Signatures. Step & Repeat can also be applied to predefined templates.
- Multiple layouts, enabling more than one matrix cell per job, for more efficient use of printed substrate when cuts are not rectangular.

The PPML format does not support substrates and finishing functions. These functions are available only in the JLYT imposition options.

The PPML features and imposition are supported on the HP Indigo press 5000 and HP Indigo production flow.

Impose a job (referred to as imposition) after a QuarkXPress document is designed and all the personalization channels that you need are added to the job.

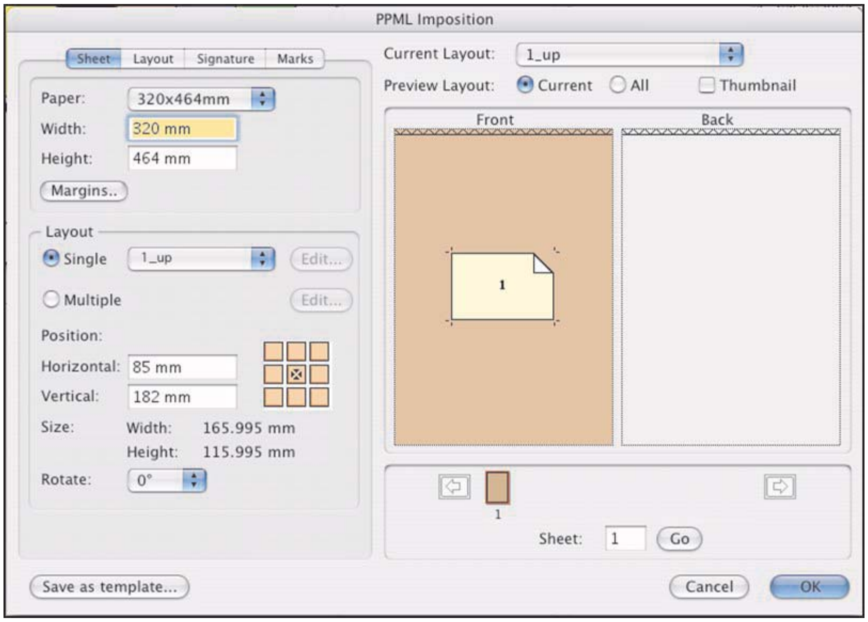
You must impose a job again if pages were added or deleted from the QuarkXPress document.

Imposition is a mandatory step in the YTD workflow, and must be performed before the output file can be created.

Note YTD supports imposition of up to 256 spreads per job.

If not defined by the user, a default imposition of single QuarkXPress page centered on a spread will be used.

Click **Yours Truly**, **Imposition,** and **PPML**. The *PPML Imposition* window appears.



PPML Imposition

Paper: 320x464mm
Width: 320 mm
Height: 464 mm

Layout
● Single 1_up
○ Multiple

Position:
Horizontal: 85 mm
Vertical: 182 mm
Size: Width: 165.995 mm
Height: 115.995 mm
Rotate: 0°

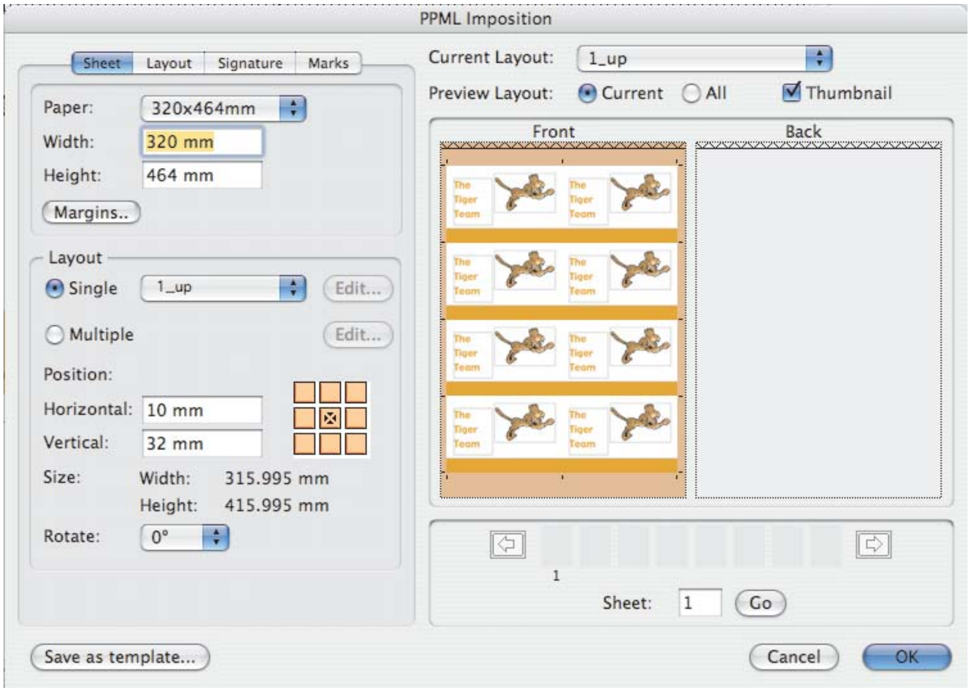
Current Layout: 1_up
Preview Layout: ● Current ○ All □ Thumbnail

Sheet: 1

Figure 5-19. ***PPML Imposition*** **window**

The *PPML Imposition* window contains two main panes:

- The tabs pane on the left contains tabs that give access to different sets of parameters that affect the imposition and printed spreads.
- The preview pane on the right contains a graphical representation of the document pages as they appear on the printed spreads. This graphical representation changes as the parameters in the tabs pane are changed. Select the **Thumbnail Preview** box to enable thumbnails. See Figure 5-20.



PPML Imposition

Paper: 320x464mm
Width: 320 mm
Height: 464 mm

Layout
● Single 1_up
○ Multiple

Position:
Horizontal: 10 mm
Vertical: 32 mm
Size: Width: 315.995 mm
Height: 415.995 mm
Rotate: 0°

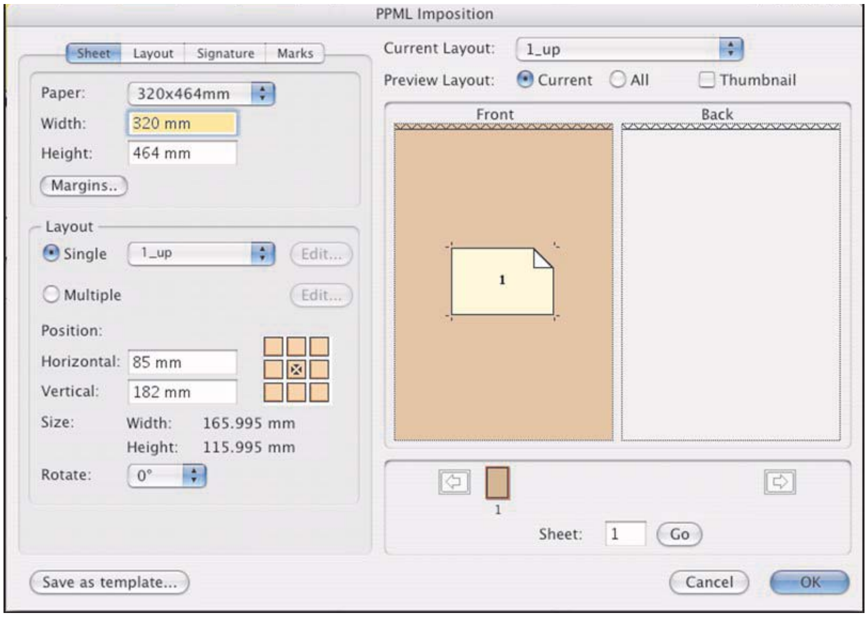
Current Layout: 1_up
Preview Layout: ● Current ○ All ☑ Thumbnail

Sheet: 1

Figure 5-20. *PPML Imposition* window - thumbnail preview

Sheet tab

1 Click the **Sheet** tab.



Current Layout: 1_up

Preview Layout: ● Current ○ All □ Thumbnail

Paper: 320x464mm

Width: 320 mm

Height: 464 mm

Layout

● Single 1_up

○ Multiple

Position:

Horizontal: 85 mm

Vertical: 182 mm

Size: Width: 165.995 mm

Height: 115.995 mm

Rotate: 0°

Sheet: 1

Figure 5-21. Sheet tab - *PPML Imposition* window

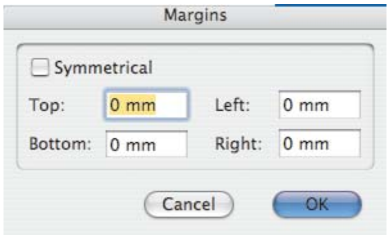
2 From the **Paper** drop-down menu, select one of the defined sizes. If the size you want does not appear in the list, select **Custom** from the list and define a new paper size (**Width** and **Height**).

3 Click **Margin** to open the *Margin* window and define the margins of the paper (spread).

a Type the margin sizes in the **Top**, **Bottom**, **Left**, and **Right** fields .

b Check **Symmetrical** if you want symmetry of top/bottom and left/right margins.

Note The page position cannot go beyond the spread margins and borders..



□ Symmetrical

Top: 0 mm Left: 0 mm

Bottom: 0 mm Right: 0 mm

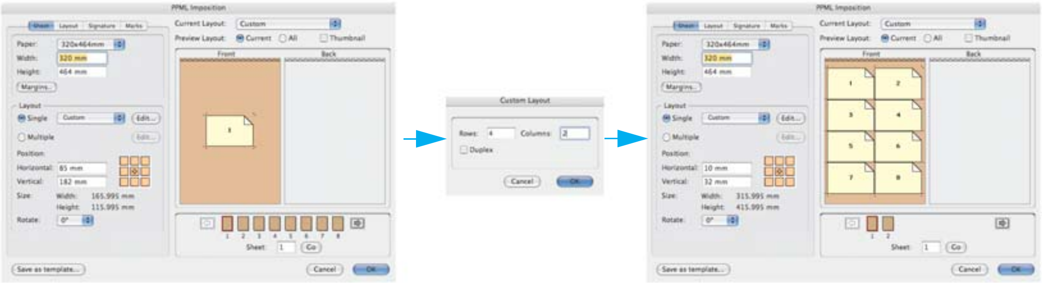
Figure 5-22. Margins window

4 Select the layout of the sheet. Choose **Single** layout to create a single layout on the sheet or **Multiple** layout to create multiple layouts on a sheet.

a When you select **Single** layout, you can:

- Select a predefined imposition from the drop-down list.

Select **Custom** from the drop-down list. The **Edit** button becomes active. Click **Edit** to open a window in which you can define sequential imposition by typing the number of rows and columns, and checking duplex if needed. Sequential imposition is for multiple QuarkXPress documents (Figure 5-23). .



PPML Imposition

Current Layout: Custom

Preview Layout: ● Current ○ All □ Thumbnail

Paper: 320x464mm

Width: 320 mm

Height: 464 mm

Layout: ● Single Custom ○ Multiple

Position: Horizontal: 85 mm Vertical: 182 mm

Size: Width: 165.995 mm Height: 115.995 mm

Rotate: 0°

Sheet: 1

Custom Layout

Rows: 4 Columns: 2

□ Duplex

PPML Imposition

Current Layout: Custom

Preview Layout: ● Current ○ All □ Thumbnail

Paper: 320x464mm

Width: 320 mm

Height: 464 mm

Layout: ● Single Custom ○ Multiple

Position: Horizontal: 10 mm Vertical: 32 mm

Size: Width: 315.995 mm Height: 415.995 mm

Rotate: 0°

Sheet: 1

Figure 5-23. Sequential imposition

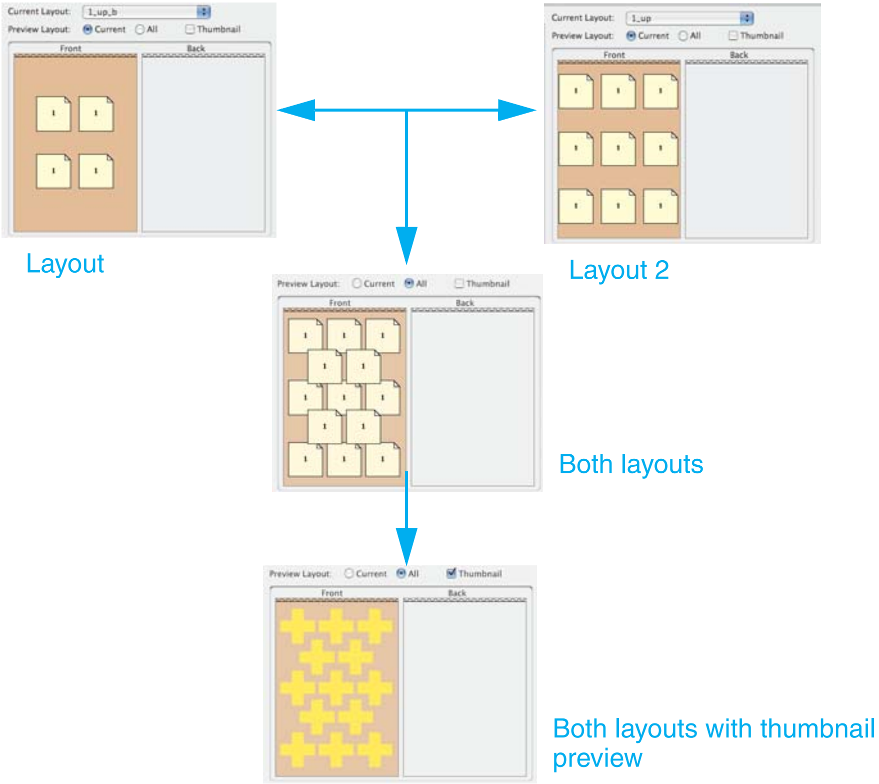
b When you select **Multiple** layout, click **Edit**. The *Multiple Layout* window opens

i Using the arrow in the window, add or remove imposition layouts as needed.

ii Click **OK** to close the *Multiple Layout* window.

c In the *PPML Imposition* window, **Current Layout** menu, select the desired layout and edit it as necessary.

In **Preview Layout**, select **All** to see all the layouts simultaneously, and check **Thumbnails** to see thumbnails of the layouts.



Current Layout: 1_up_b

Preview Layout: ● Current ○ All □ Thumbnail

Current Layout: 1_up

Preview Layout: ● Current ○ All □ Thumbnail

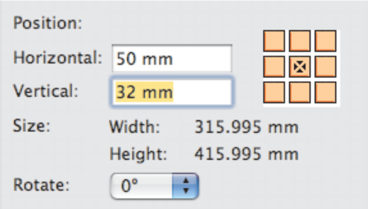
Preview Layout: ○ Current ● All □ Thumbnail

Preview Layout: ○ Current ● All ☑ Thumbnail

Figure 5-24. Multiple layout preview examples

5 You can align the block of pages to nine different positions within the margins: top/middle/bottom and left/center/right.

Type horizontal and vertical position values in the respective **Position** fields, or use the positioning keys (shown below) to align the pages.



Position:
Horizontal: 50 mm
Vertical: 32 mm
Size: Width: 315.995 mm
Height: 415.995 mm
Rotate: 0°

Figure 5-25. Positioning keys

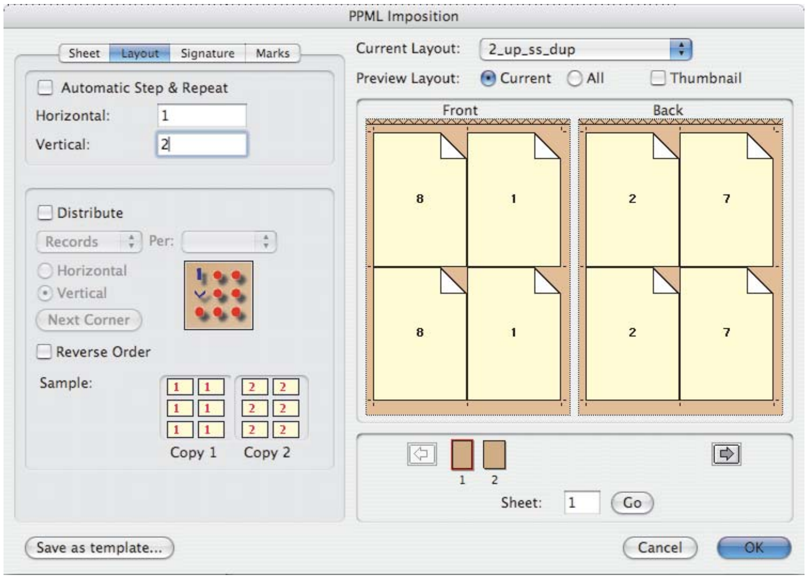
Note The Width and Height displayed in the Layout pane are based on the sum of the pages' widths and heights, and on the sizes of the crop marks, bleed, and gutters.

The default values of the **Paper Size** and **Margins** fields can be modified. Additionally, a new paper can be added to the list. See "Changing paper parameters" on page 140.

6 You can rotate the entire layout by typing a rotation angle value in the **Rotate** field.

Layout tab

1 Click the **Layout** tab.



PPML Imposition
Current Layout: 2_up_ss_dup
Preview Layout: ● Current ○ All ☐ Thumbnail
☐ Automatic Step & Repeat
Horizontal: 1
Vertical: 2
☐ Distribute
Records Per:
○ Horizontal
● Vertical
☐ Reverse Order
Sample:
Sheet: 1

Figure 5-26. Layout tab- *PPML Imposition* window

2 Select the option for the type of imposition you want:

a **Automatic Step & Repeat**:
Check this box to cause the first page to be repeated many times on the first spread, the second page to be repeated many times on the second spread, and so on.

b **Manual Step & Repeat:**
Select the relevant values in the **Horizontal** and **Vertical** repeats in the relevant field

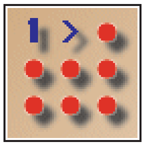
c **Distribute**:
Check this box to activate the layout distribution options. Select the distribution method from the drop-down menus:

- **For Personalization Jobs**:
  - **Records** Per **Spread** assigns the channel cycles per spread. The same assignment order is applied to each spread.
  - **Records** Per **Job** assigns the channel cycles per job. The assignment is applied to the entire job, across all the spreads.
  - **Records** Per **Line** assigns the channel cycles per line across the entire job.

  Click **Next corner** to position the first channel cycle in a page that is a different corner of the spread.
- **For Imposition Jobs**:
  - **Pages** Per **Spread** assigns the document pages per spread. The same assignment order is applied to each spread.
  - **Pages** Per **Job** assigns the document pages per job. The assignment is applied to the entire job, across all the spreads.
  - **Pages** Per **Line** assigns the document pages per line across the entire job.

  Click **Next corner** to position the first page in a different corner of the spread.
- Choose the assign direction by selecting the **Horizontal** or **Vertical** radio button.
- Check **Reverse Order** to reverse the order of the assign direction.

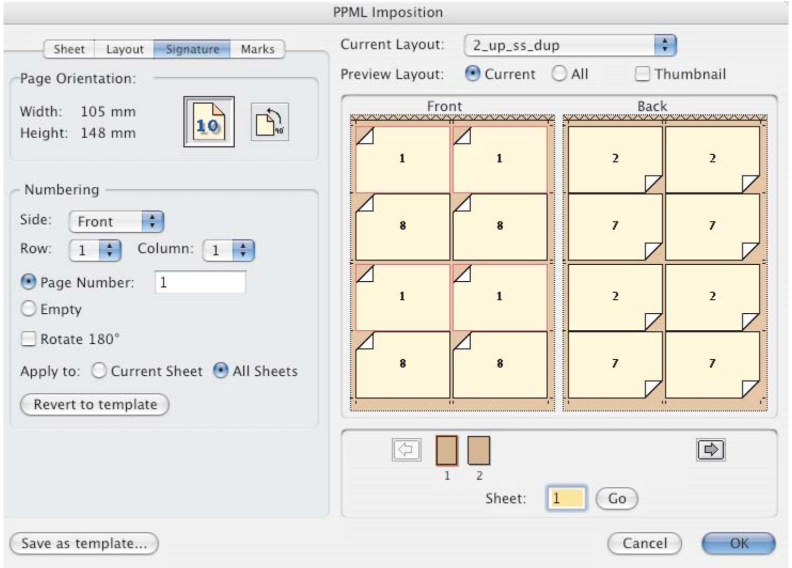
Note The positioning icon shows the position of the first channel cycle or the first page and the direction of the subsequent cycles or pages.



Signature tab

In the **Signature** tab numbering pane for a multi-page QuarkXPress document, you can change the page orientation, replace a page by a specific page on the spread, and define the number of sheets in the imposition.

1 Click the **Signature** tab.



PPML Imposition

Current Layout: 2_up_ss_dup

Preview Layout: ● Current ○ All □ Thumbnail

Page Orientation:

Width: 105 mm
Height: 148 mm

Numbering

Side: Front

Row: 1 Column: 1

● Page Number: 1

○ Empty

□ Rotate 180°

Apply to: ○ Current Sheet ● All Sheets

Sheet: 1

Figure 5-27. Signature tab - *PPML Imposition* window

2 To change the page orientation for all the pages, click on the button in the Page Orientation pane.

The page dimensions are shown in the Width and Height fields, and the orientation is displayed in the icon to the left of the button.

3 To change one or more of the pages, do the following:

a In the Numbering pane, select the page that you want to change by selecting the appropriate values in the Side, Row and Column drop-down menus. The selected page(s) in the preview pane are surrounded by a thin red frame.

b Select the type of change:

- To replace the page with a specific page, click the **Page Number** radio button and type the page number.
- To replace the page with an empty page, click the **Empty** radio button.
- To rotate the page 180^{o}, select the **Rotate 180** check box.

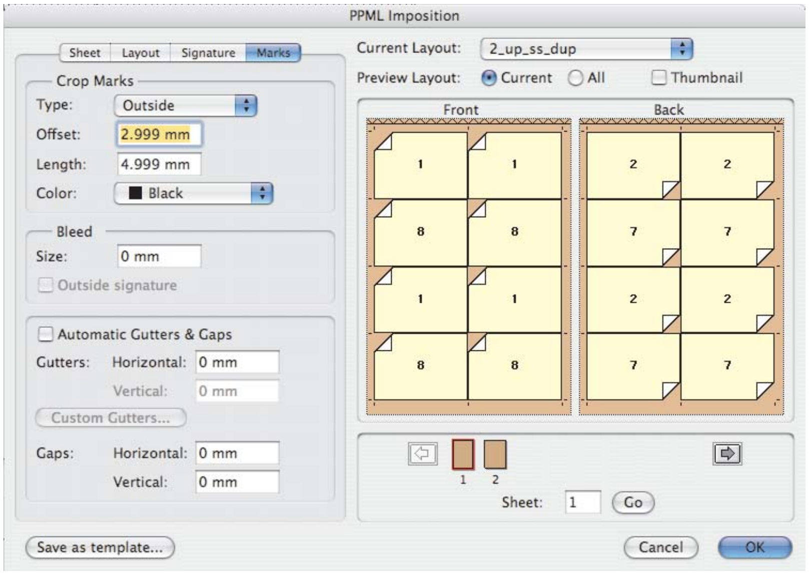
c Select the relevant radio button:

- Apply to current sheet
- Apply to All sheets

4 To revert to the original template signature, click **Revert to template.**

Marks tab

1 Click the **Marks** tab.



PPML Imposition

Crop Marks

Type: Outside

Offset: 2.999 mm

Length: 4.999 mm

Color: ■ Black

Bleed

Size: 0 mm

☐ Outside signature

☐ Automatic Gutters & Gaps

Gutters: Horizontal: 0 mm

Vertical: 0 mm

Gaps: Horizontal: 0 mm

Vertical: 0 mm

Current Layout: 2_up_ss_dup

Preview Layout: ● Current ○ All ☐ Thumbnail

Sheet: 1

Figure 5-28. Marks tab - *PPML Imposition* window

2 Select the appropriate settings in the Crop Marks pane:

a In the **Type** drop-down menu, select one of the following:

- **None**, for no crop marks.
- **Outside**, for crop marks on the outside perimeter of the block of pages.
- **Every Signature**, for crop marks on the outside perimeter of every signature.
- **Every Page**, for eight crop marks on each page.

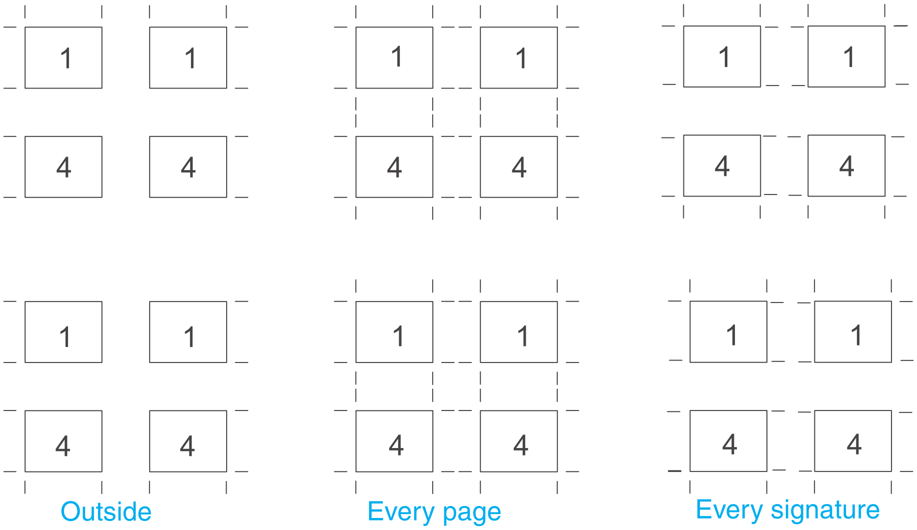


Figure 5-29. Types of crop marks

b In the **Offset** field, type the distance between the end of the page and the beginning of the crop mark.

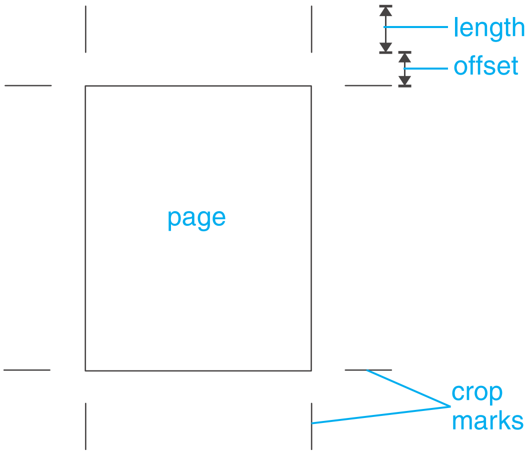


Figure 5-30. Example of offset

c In the **Length** field, type the length of the crop mark.

d In the **Color** drop-down menu, select the color of the crop marks. You can select **Registration** to have the crop marks printed in all the separations of the job.

3 Select the appropriate settings in the Bleed pane:

a In the **Size** field, type the size of the bleed area.

b If you want the bleed area to be outside the block of pages only, select the **Outside only** check box. If this check box is not selected, the bleed will be in the gutters too.

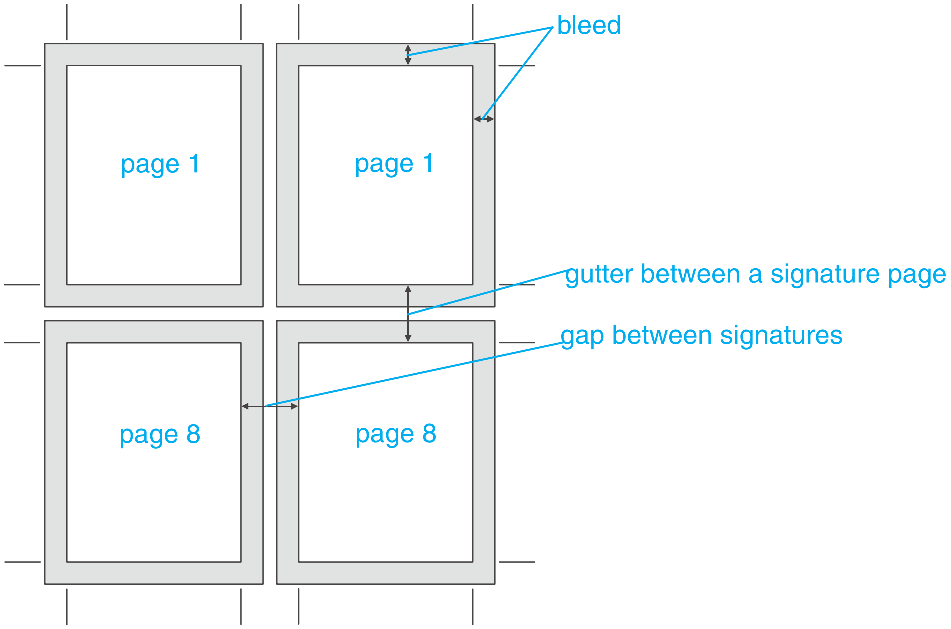


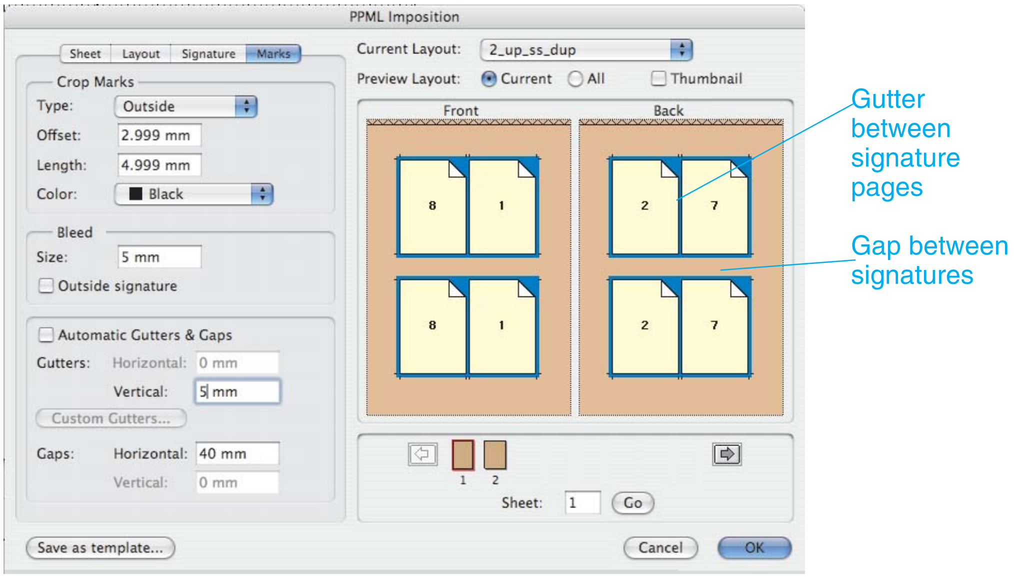
Figure 5-31. Bleed, gutter, and gap

Note The default values of the Crop Marks and Bleed fields can be changed. See "Changing marks parameters" on page 143.

4 Select the appropriate settings in the Gutters pane:

a In the **Vertical** and **Horizontal** fields, type the size of the space (gutter) between the columns (vertical) and rows (horizontal) of the pages.

If the vertical or horizontal gutter size measures less than the bleed size, there will be no bleed in the column's or row's gutter:



PPML Imposition

Crop Marks
Type: Outside
Offset: 2.999 mm
Length: 4.999 mm
Color: Black

Bleed
Size: 5 mm
□ Outside signature

□ Automatic Gutters & Gaps
Gutters: Horizontal: 0 mm
Vertical: 5 mm

Gaps: Horizontal: 40 mm
Vertical: 0 mm

Current Layout: 2_up_ss_dup
Preview Layout: ● Current ○ All □ Thumbnail

Sheet: 1

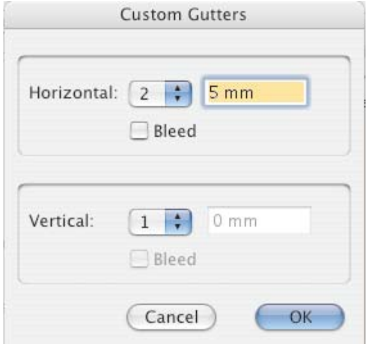
Figure 5-32. Bleed, gutters, gaps

b Check the **Automatic Gutters & Gaps** check box to automatically set the gutter size to twice the sum of the crop marks offset plus length or to twice the bleed size, whichever is greater. If the crop mark type is None or Outside, its offset and length do not affect the gutter calculations. Similarly, if the bleed is Outside only, its size does not affect the gutter calculations.

Examples (numbers are in mm):

Offset	Crop mark length	Bleed size	Automatic gutter size
5	2	6	14
5	2	11	22
5	2	11 Outside	14
5	2 Outside	6	12
5	2 Outside	11 Outside	0

c To give different values to different vertical and horizontal gutters and bleed, click **Custom**. The *Custom Gutters* window appears.



Custom Gutters

Horizontal: 2 5 mm

□ Bleed

Vertical: 1 0 mm

□ Bleed

Figure 5-33. Custom gutters

In the Vertical Gutters and Horizontal fields, do the following:

i In the number drop-down menus, select the gutter number, left to right and top to bottom.

ii Type the size of the gutter in the field next to the gutter number.

iii To clear the bleed in the selected gutter, clear the **Bleed** check box.

If the gutters or bleed are customized, the word `*custom` appears in the Gutters pane and red arrows appear in the preview pane.

d Define the horizontal and vertical **Gap** values. The gap is the distance between the layout signatures. A gap value of 0 produces layouts that are touching each other.

Concluding the PPML imposition

1 Click **OK**.

2 If you want to add spread elements, proceed to "Adding spread elements" on page 118.

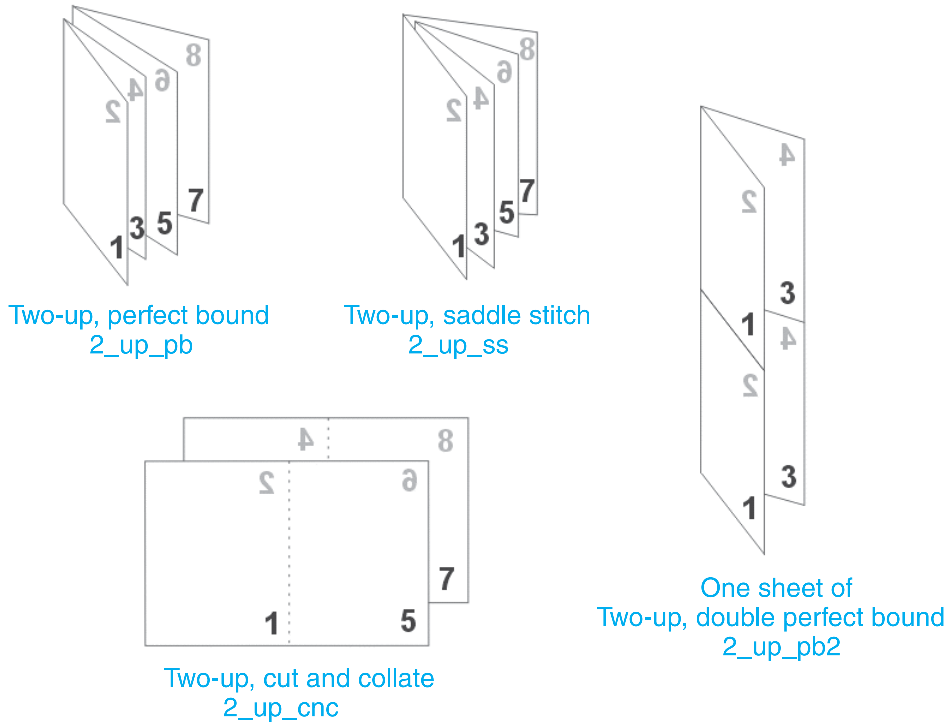
Otherwise, proceed to "Creating the output file" on page 119.

Saving the imposition as a template

Create your own imposition template at any time during imposition by clicking **Save as Template** in any of the tab pages. The currently applicable imposition values are saved as a new template.

Templates can be generic or non-generic. Generic templates are applicable to documents with any number of pages or page sizes. Non-generic templates are applicable only to documents that have the same number of pages and page sizes as in the QuarkXPress document used to create the template. Generic template names are **bold** while non-generic templates are plain.

YTD-supplied template examples appear in Figure 5-34.



Two-up, perfect bound
2_up_pb

Two-up, saddle stitch
2_up_ss

One sheet of
Two-up, double perfect bound
2_up_pb2

Two-up, cut and collate
2_up_cnc

Figure 5-34. Examples of YTD-supplied templates

Create an imposition template

Create a JLYT or PPML imposition template for current or future use in imposing QuarkXPress documents. Create a JLYT imposition for sending to the HP Indigo RIP or the HP Indigo press where it can be used for imposition.

To create an imposition template:

1 Create an empty QuarkXPress document with the following characteristics:

- Document size required for imposition.
- Number of pages required to fill at least one sheet.

For example, for a two-up duplex job, select a page size that is small enough that two pages can fit on the spread, and create the document with at least four pages, two for each spread of the sheet.

2 In the YTD menu, select **JLYT Imposition or PPML Imposition**.

3 Position the document pages on the spread as required. You can use all the options described in this chapter.

4 After definition, select **Save as template**. Save the template using a descriptive name.

5 Verify that the new imposition template appears in the template list. When the template name appears in bold, the template is generic, and applicable to all QuarkXPress documents, regardless of the number of their pages.

● JLYT imposition template files which are saved in the *Mac_Disk:Applications:QuarkXPress 7:XTensions:Yours Truly:Imposition* folder.

● PPML imposition template files which are saved in the *Mac_Disk:Applications:QuarkXPress 7:XTensions:Yours Truly:PPML Imposition* folder.

Adding JLYT imposition templates to the press

The YTD-supplied JLYT imposition templates are also found on the press. You can use YTD to create and transfer a new JLYT imposition template to the press to be used with the imposition tool or while RIPping.

Note The press software supports only JLYT imposition templates. It does not support YTD JLYT imposition templates that contain:
● bleed outside only (see "Marks tab" on page 92)
● bleed that is less than the gutter
● custom gutter definition (see Figure 5-12. on page 95)

These templates must not be sent to the press.
The press software does not support PPML imposition templates.

1 If the specific template that you need does not exist, create it in YTD using JLYT imposition (see "Imposing a job using JLYT imposition" on page 87).

2 If the imposition template exists in the JLYT template list, open the Imposition folder through the folder path: *Mac_Disk:Applications:QuarkXPress 7:Xtensions:Yours Truly:Imposition*.

3 In the Imposition folder, locate the template that you need to transfer, in the Name column.

4 To copy the JLYT template to the press for use with the imposition tool, do the following for all presses except for the HP Indigo press 5000:

a Open the HP Indigo press shared volume.

b Open the Press, input, and lan folder. PressName

c Drag the template from the Imposition folder to the lan folder.

The template icon appears in the lan folder for a few seconds, during which time it is sent to the press.

When the template icon disappears, this indicates it was successfully sent to the press. It can now be used on the press by the Imposition tool.

5 To copy the JLYT template to the press for use while RIPing, do the following for all presses but the HP Indigo press 5000:

a Open the HP Indigo press shared volume.

b Open the RIP, input, and 4colors folder. PressName

c Drag the template from the Imposition folder to the 4colors folder. The template icon appears in the 4colors folder for a few seconds, during which time it transfers to HP Indigo RIP.

When the template icon disappears, this indicates that it successfully transferred to the HP Indigo RIP on the press. To use the template while RIPping, the HP Indigo RIP Service must restart on the press.

6 For the HP Indigo Press 5000, use the following steps for using imposition template with either the press imposition tool or while RIPping:

a Open the HP Indigo press shared volume.

b Copy the imposition template file to the shared volume in any location.

c At the press, open the **Admin** menu, select **RIP Imposition**

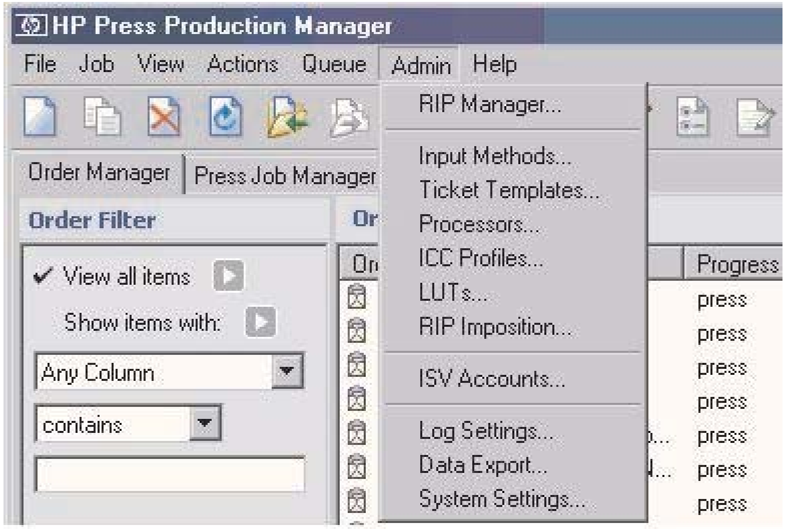
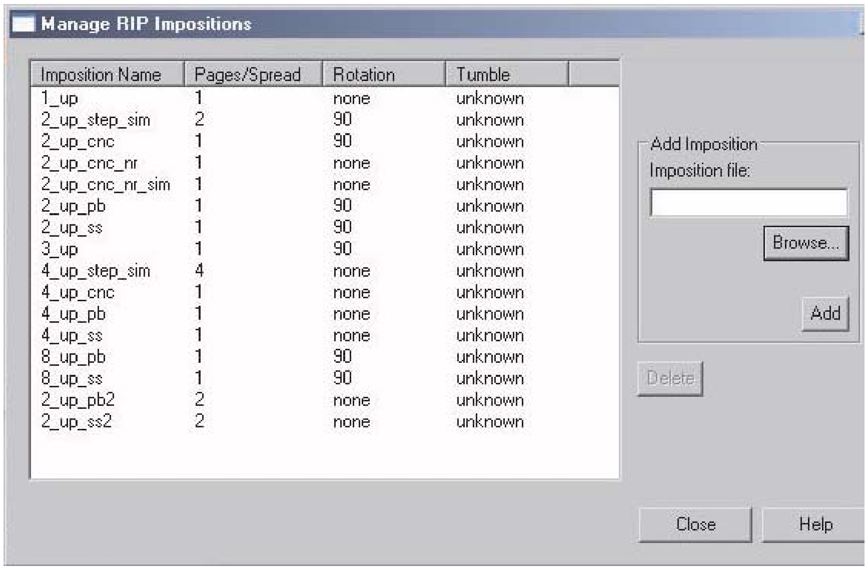


Figure 5-35. HP Indigo Press 5000 *Admin* window for RIP imposition

d In the window that opens, browse to locate the imposition file, and select **Add**.



Imposition Name	Pages/Spread	Rotation	Tumble
1_up	1	none	unknown
2_up_step_sim	2	90	unknown
2_up_cnc	1	90	unknown
2_up_cnc_nr	1	none	unknown
2_up_cnc_nr_sim	1	none	unknown
2_up_pb	1	90	unknown
2_up_ss	1	90	unknown
3_up	1	90	unknown
4_up_step_sim	4	none	unknown
4_up_cnc	1	none	unknown
4_up_pb	1	none	unknown
4_up_ss	1	none	unknown
8_up_pb	1	90	unknown
8_up_ss	1	90	unknown
2_up_pb2	2	none	unknown
2_up_ss2	2	none	unknown

Figure 5-36. Adding an imposition template file

The imposition template file will then be available for use from the press imposition tool or while RIPping.

Adding spread elements

This section describes how to add elements (text, graphics, and images) to the spread. The elements may be standard QuarkXPress elements and/or YTD variable channels, depending on the job type: personalization or imposition only.

You can add spread elements only if you have imposed the job beforehand.

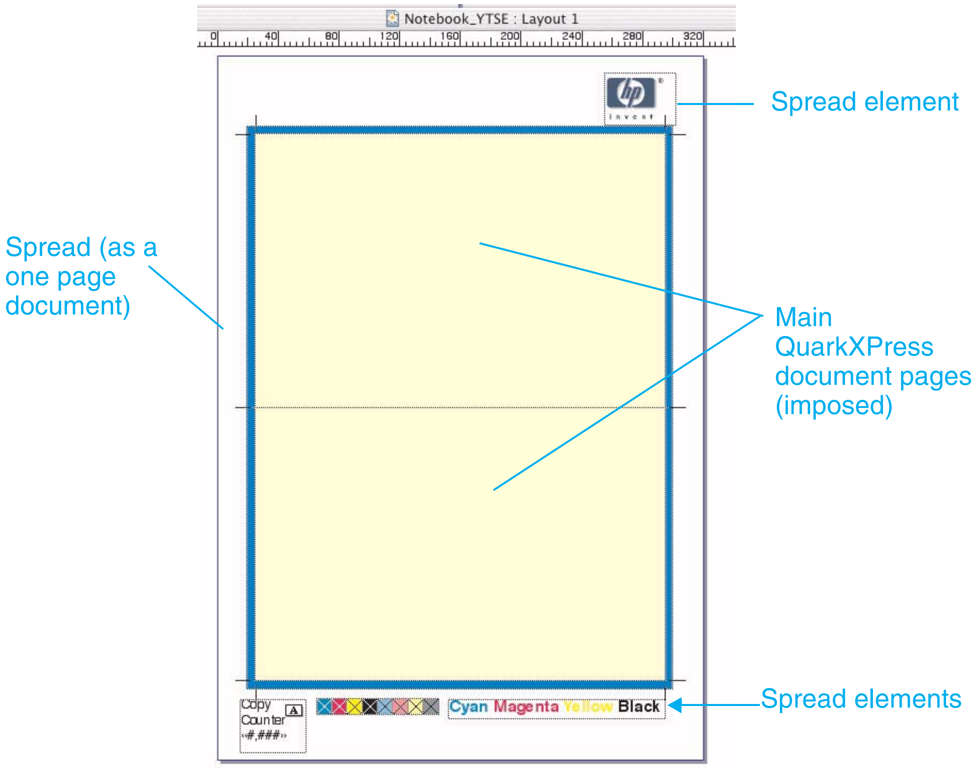


Figure 5-37. Spread elements

In the QuarkXPress menu bar, click **Yours Truly** and **Spread Elements**.

The spread appears as a one-page QuarkXPress document. Displayed on the spread are the pages of the main QuarkXPress document, as yellow-colored pages, in the layout defined in imposition.

The name of the spread element document is xxx_YTSE (**Y**ours **T**ruly **S**pread **E**lement), where xxx is the original document's name.

You can add spread elements to any job. Spread elements can be:

- Any QuarkXPress box/path
- YTD text channels (only for personalization jobs, using database fields)
- YTD indexing

Save and close the YTSE document when you are done adding spread elements. This returns you to the main QuarkXPress document.

Spread elements appear in both the front and back spreads of a duplex job if the imposition is centered. Spread elements appear only on the front spread if the imposition is not centered.

Proceed to "Creating the output file" on page 117.

6 Creating the output file

This chapter contains the following topics:

- Overview
- Creating an output file
- Exporting as a PDF file
- Mounting the press shared volume
- Sending the output JLYT or PPML file to the press
- Sending the PDF output files to the press

Overview

This chapter describes how to create an output JLYT, PPML, or PDF file for the QuarkXPress document, how to send or print this output file to the HP Indigo press, and how to export PDF files.

The personalization output JLYT and PPML files can be either a Job output file or a Template output file.

- The Job output file contains an embedded DB report file. It may also contain embedded elements such as variable images and/or variable text that were processed to PostScript by the YTD software. If the QuarkXPress pages contain background elements, the output file contains embedded PostScript elements for the background as well. The output file can be printed according to its embedded DB file only.
- The Template output file does not contain an embedded DB report file nor any variable data processed to PostScript. If a background exists, the output file does contain the background data in PostScript format. When creating a Template output file, you can print the file at the press many times, each time using a different DB report file.

A QuarkXPress document that contains at least one personalization channel that is not suitable for creating a SNAP job always results in a Job output file (see list of acceptable SNAP channel attributes on page 26).

If the QuarkXPress document was designed as a SNAP job using the **Force Snap** check box, you can decide whether to create a Job or a Template output file. That is, whether the DB report file is embedded in the output file or not.

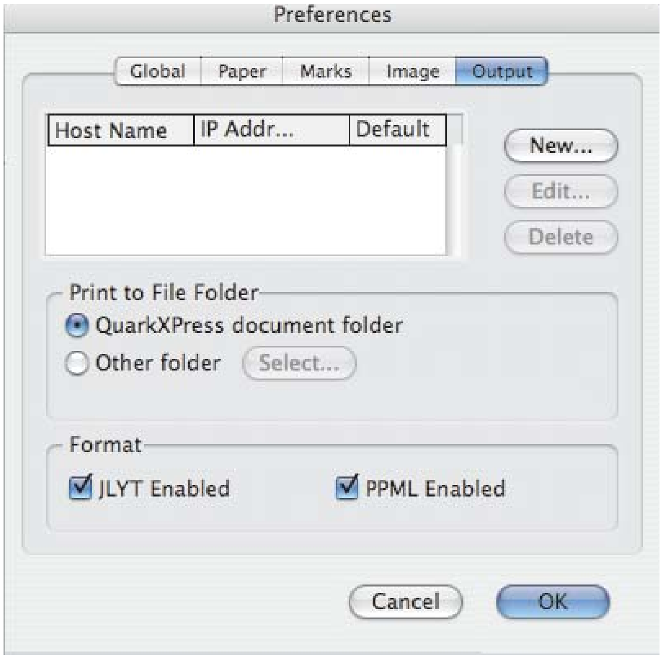
A PDF output file can be created for a personalized or a non-personalized job.

Note Naming conventions for files on the Macintosh and PC/Windows are different. Since YTD jobs are sent to the press' computer, which is a PC/Windows system, all file names in YTD (including job names and document names) must conform to the PC/Windows naming conventions. This means that file names can be composed of all alphanumeric and special characters except \ / * : ? | < >.

Note Only the HP Indigo press 5000 and HP Indigo Production Flow support PPML output. PPML format may be used to create both output job files and template job files.

Creating an output file

You can create an output JLYT file, a PPML file, or a PDF file for the QuarkXPress document. JLYT is the default format. To enable YTD to create PPML output files, you must select the check box in the *Preferences* window prior to creating the job.



Print to File Folder

● QuarkXPress document folder

○ Other folder

Format

☑ JLYT Enabled ☑ PPML Enabled

Figure 6-1. Enabling PPML in the *Preferences* window

In the QuarkXPress menu bar, click **Yours Truly** and **Create Job**.

If you did not save the most recent changes made to the document, you must save them now.

If you have enabled both PPML and JLYT output formats, a window will open with the options to select JLYT or PPML. Select the format you want for the job.

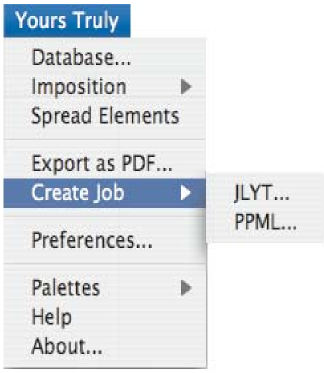
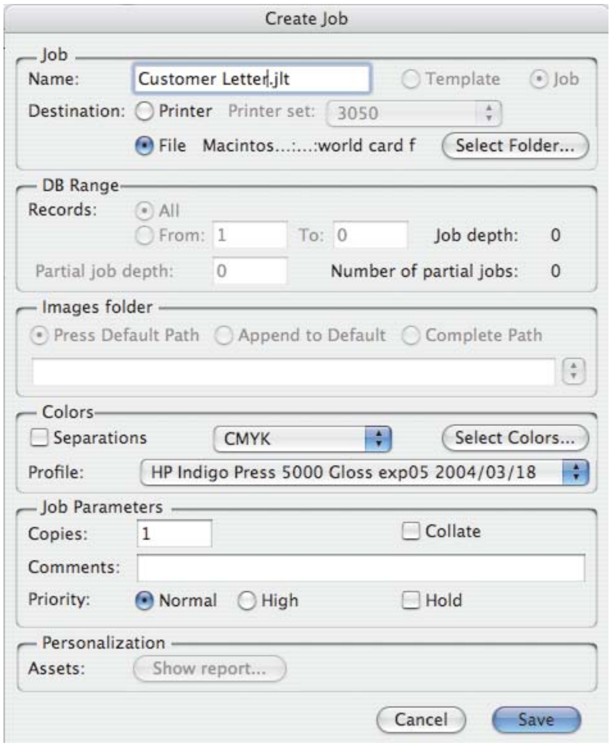


Figure 6-2. Create Job options

The *Create Job* window appears.



Create Job

Job

Name: Customer Letter.jlt ○ Template ● Job

Destination: ○ Printer Printer set: 3050

● File Macintos...:...:world card f

DB Range

Records: ● All

○ From: 1 To: 0 Job depth: 0

Partial job depth: 0 Number of partial jobs: 0

Images folder

● Press Default Path ○ Append to Default ○ Complete Path

Colors

□ Separations CMYK

Profile: HP Indigo Press 5000 Gloss exp05 2004/03/18

Job Parameters

Copies: 1 □ Collate

Comments:

Priority: ● Normal ○ High □ Hold

Personalization

Assets:

Figure 6-3. ***Create job* window**

Enter information for the following fields:

1 The **Name** field in the Job pane should contain the name of the output file that will be sent to the press for printing. By default, this is the QuarkXPress layout name. If you want to preserve job name consistency (on the Macintosh and press), do not change the job name in this field.

2 The **Template** and **Job** radio buttons are available if you prepared your document using the **Force Snap** option (see "Creating template jobs" on page 25). Select the **Template** radio button to create a template job, select the **Job** radio button to create a non-template job.

The **Template** and **Job** radio buttons are disabled (gray) and **Job** is selected when:

- At least one channel contains parameters that are not suitable for template jobs
- It is an imposition only job (without personalization), or
- Destination is set to Printer

The **Template** and **Job** radio buttons are enabled and **Job** is selected by default when:

- All channels are Snap channels, but the **Force Snap** check box in the *Channels* palette is left unchecked.

Note

When creating a personalization template output file, you can print the file at the press several times, each time using a different DB report file.

When creating a personalization job output file, the output will contain an embedded database that will be printed with the job only.

3 **Destination** radio buttons:

- To create and send the output file to the press for printing, select the **Printer** radio button. Then select the press from the **Printer set** drop-down menu. You can define printers from the Preferences window (see "Changing output parameters" on page 147). The output JLYT or PPML file is not saved on the Macintosh.
- To create an output JLYT or PPML file and write it to the Macintosh, select the **File** radio button. The default destination for the file is the same location as the QuarkXPress document. You can select a different destination using **Select Folder**. You change the default destination in the *Preferences* window (see "Images folder" on page 146).

4 **DB Range** pane:

For Job files:

- Select the **All** radio button in the DB range pane if you want to print all the records in the DB report file.
- To restrict the range of DB report file records to print, select the **From** radio button. In the **From** and **To** fields, type the start and end record numbers that you want to print. Valid values are 1 to 32,000. This option is applicable to Job (non-template) output files only.
- If you want to divide the output into more than one job, type the number of records for each job in the **Partial job depth** field. In case the job contains at least one non-template channel, YTD produces a multi-page PostScript file that is embedded in the output file.

For Non-Snap jobs:

- QuarkXPress has a 2,000-page limit per document. Therefore, if the DB report file contains more than 2,000 records, YTD will not be able to create a multi-page QuarkXPress document for that image channel. To bypass this limitation, the **Partial job depth** field allows you to divide the output into more than one job, each job with no more than 2,000 QuarkXPress pages.

5 In the Images folder pane, choose the path on the press of the variable images files as follows:

- Select the **Press Default Path** radio button if the job accesses its image files at the press default path (usually S:\JOBS\PRESS\IMAGES or for HP Indigo press 5000 S:\JOBS\4COLORS\IMAGES). This radio button is selected by default.
- Select the **Append to Default** radio button to append a sub-folder path to the press default path.

  Type the sub-folder path in the field. Alternatively, select a previously saved sub-folder path from the field's drop-down menu. To save the sub-folder path for future use, select **Add to list** from the field's drop-down menu.
- Select the **Complete Path** radio button for a different destination.

  Type the complete path in the field. For example: `S:\janprojects\images`, where S is the drive letter. Alternatively, select a previously saved complete path from the field's drop-down menu. To save the complete path for future use, select **Add to list** from the field's drop-down menu.

Note To edit the path list, see the *Preferences* window (see "Images folder" on page 146).

6 In the Colors pane, choose the printing option and separation set as follows:

- To print composite colors, leave the **Separations** check box cleared.

- To print separations, check the **Separations** check box. You will need to print separations if you have overprinting colors.
- To print HP IndiChrome colors, select **HP IndiChrome** in the drop-down list.
- To print HP IndiChrome colors with additional spot colors, do the following:

  i Select **HP IndiChrome** in the drop-down list.

  ii Click **Select Colors.** The *Custom Colors* window opens..

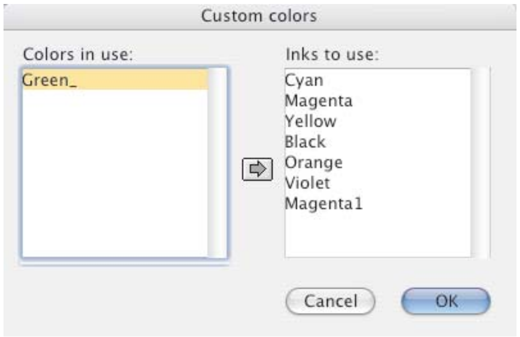


Figure 6-4. ***Custom colors*** **window**

  iii Select the spot color in the *Colors in use* column and click the arrow to move them to the *Inks to use* column. The colors listed in the *Inks to use* column are the colors used by the HP Indigo press.

  When HP IndiChrome is selected, by default, CMYKOVM1 is selected in the *Inks to use* column. You can only add one additional spot color.

  iv Click **OK** when you finish selecting the spot color. The *Custom colors* window closes.

- To print in grayscale, select **Black** in the drop-down list. The job prints using black ink only. RGB information is converted to gray scales, while with CMYK data, the CMY information is omitted, and only the black is printed.
- If you need to print more than CMYK colors (composite or separations), do the following:

  i Click **Select Colors**.

  The *Custom colors* window appears.

  ii Select the spot color in the *Colors in use* column and click the arrow to move them to the *Inks to use* column. The colors listed in the *Inks to use* column are the colors used by the HP Indigo press.

  iii Click **OK** when you finish selecting the spot colors.

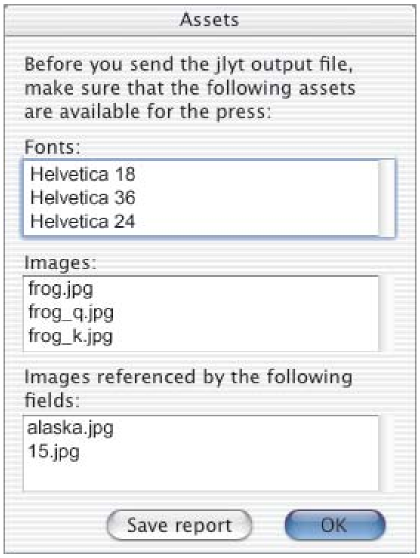
- ICC profiles are supported on HP Indigo press 5000 only. The default ICC profile selected for the press output profile in YTD is identical to the default setting of the press: Suitable for glossy paper. Additional ICC profiles are available in the *Colors, Profile* list in the *Create Job* window. ICC parameters are ignored when the output file is sent to any press other than HP Indigo press 5000.

7 In the **Job Parameters** pane:

- for imposition only jobs, in the **Copies** field, type the number of copies of the job that you want printed. The default value for this field is defined in the *Preferences* window (see "Changing global parameters" on page 138).

- Select the **Collate** check box if you want the job's multiple sheets and copies to be collated.
  For example: If your job has three sheets and three copies, the collated sheets will be printed 123123123 rather than 111222333 when not collated.
- In the **Comments** field, type any comments that you want associated with this job. These comments appear in the Job Manager listing of the job at the press.
- **Priority** radio buttons: select **Normal** or **High** priority. This setting appears as a priority number in the Job Manager listing of the job at the press.
- Select the **Hold** check box if you want the output job on the press to remain in the Loaded Jobs queue instead of move to the Print Queue.

8 In the Personalization pane, click **Show report** to see the job assets that need to be present on the press. The *Assets* window appears.


Assets

Before you send the jlyt output file, make sure that the following assets are available for the press:

Fonts:
Helvetica 18
Helvetica 36
Helvetica 24

Images:
frog.jpg
frog_q.jpg
frog_k.jpg

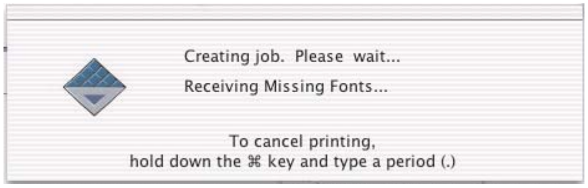
Images referenced by the following fields:
alaska.jpg
15.jpg

Figure 6-5. ***Assets*** **window**

This window lists the assets that must be available to the press before you send the output JLYT or PPML file. You can click **Save report** to save the report as a text file. The report is saved in the same folder as your QuarkXPress document. This report can be printed for reference. The *Assets* window and the report contain the following fields:

- **Fonts**: lists fonts needed for the text personalization channels. You will need to download any fonts to the press if they are not currently available to the press on which you intent to print the output file. See "SNAP fonts" on page 151 for details.
- **Images**: lists image files needed for the image personalization channels. These specific images must be available for the press by being placed in a directory that you selected in the Images folder pane of the *Create Job* window.
- **Images referenced by the following fields**: lists the fields in which image files are referenced. These image files are needed for the personalization image channels. The specific images must be available for the press by being placed in a directory that you selected in the Images folder pane of the *Create Job* window.
- Click **OK** when you finish viewing the assets.
- If your job destination is a printer, you must ensure that all the fonts and images needed for the job are available to the press before you click **Print** in the *Create Job* window.

- If your destination is printer, and any font is missing at the press, a warning appears during the job creation process.



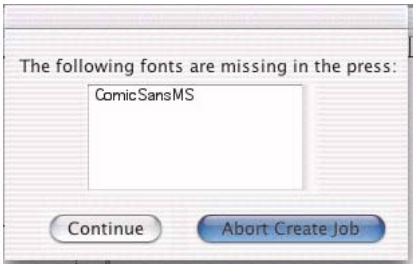
Creating job. Please wait...

Receiving Missing Fonts...

To cancel printing,
hold down the ⌘ key and type a period (.)

Figure 6-6. Missing fonts warning

The following window appears.



The following fonts are missing in the press:

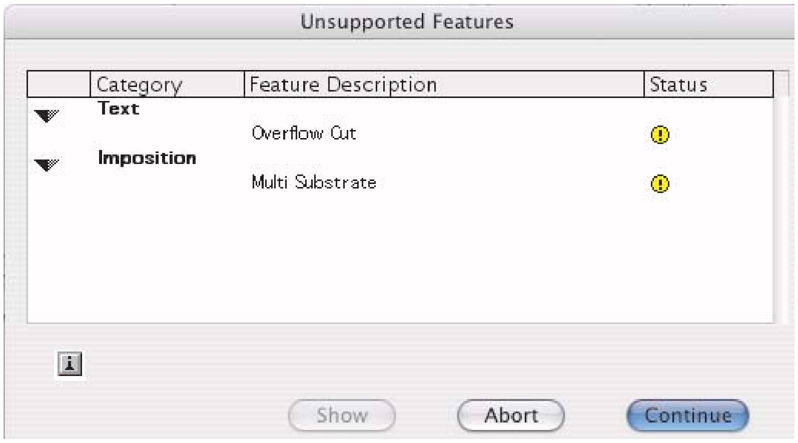
ComicSansMS

Figure 6-7. *Missing fonts* window

9 You can:

- Click **Continue** to create the job without the font. You will need to replace the font or abort the job on the press.
- Click **Abort Create Job** to abort the job creation. You can then send the missing font to the press, and then proceed to create the job.

10 If you have selected the JLYT imposition and PPML output or PPML imposition and JLYT output, certain features used in the imposition may not be supported by your chosen format. the *Unsupported Features* window appears listing the unsupported features. Yellow item in the *Unsupported Features* window may be ignored, but red items cannot be ignored by the job.



Unsupported Features

Category	Feature Description	Status
Text		
	Overflow Cut	!
Imposition		
	Multi Substrate	!

Figure 6-8. *Unsupported Features* window

If you have unsupported features, you can:

- Click **Continue** to create the job and ignore the unsupported features.
- Select an item from the *Unsupported Features* window and click **Show.** This marks the specific channel in the document.
- Click **Abort Create Job** to abort the job creation.

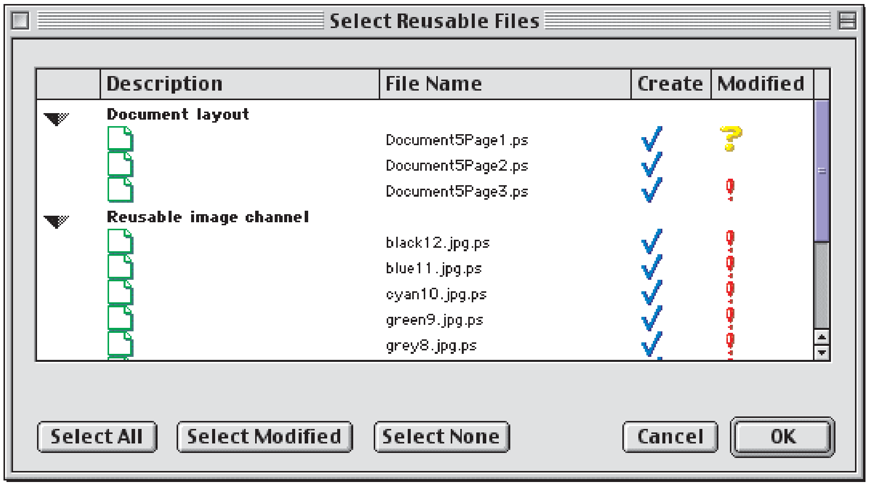
PPML does not support the following features:

- **Job**
  - External PS Channel Type
- **Image**
  - JLYT image type
  - Image mode: Bitmap/Grayscale
- **Text**
  - Overflow Cut
  - Horizontal alignment with Word Wrap
  - Baseline shift with Word Wrap
  - Track and Kern
- **Imposition**
  - Sequential Layout
  - Custom Gutters with Step & Repeat
  - Multi Substrates
  - Distribute Data with special cases (e.g. numbering tab pages manipulations)
- **Spread Element**
  - Text Channel

JLYT does not support the following features:

- Pages rule
- Layers rule
- Multiple imposition layers

11 If you selected the **Printer** radio button, the **Print** button appears in the *Create Job* window. If you selected the **File** radio button, the **Save** button appears in the *Create Job* window. Click **Print** or **Save**, whichever appears in the window.
The *Select Reusable Files* window appears.



Select Reusable Files

Description	File Name	Create	Modified
Document layout			
	Document5Page1.ps	✓	?
	Document5Page2.ps	✓	
	Document5Page3.ps	✓	!
Reusable image channel			
	black12.jpg.ps	✓	!
	blue11.jpg.ps	✓	!
	cyan10.jpg.ps	✓	!
	green9.jpg.ps	✓	!
	grey8.jpg.ps	✓	!

Figure 6-9. ***Select Reusable Files*** **window**

The Description column, contains the following:

- Under the Document layout heading are PostScript files; one PS file per main QuarkXPress document page that contains fixed data. The PS files' names appear in the File Name column as `aaaPagex.ps`, where `aaa` is the main QuarkXPress document name and `x` is the page number.
- If the job contains a Catalog document channel, then under the Reusable image channel heading are the PostScript files that will be created from the multi-page QuarkXPress catalog document. The PS files' names appear in the File Name column as `bbb.ps`, where bbb equals the name you selected when you verified the PostScript names (during "Verifying the PostScript names" on page 46).

The icons in the Create and Modified columns are as follows:

Icon	Description
✓	PS file listed in the File Name column will be created and embedded in the output file.
!	The page was never printed. This page is automatically marked ✓ in the Create column and cannot be unmarked.
?	The page may have been modified since its previous print. If you are sure that the page was not modified, clear the mark ✓ in the Create column.
blank in Modified column	The page was not modified since its previous print. If you do not want to recreate the page for printing, clear the mark ✓ in the Create column.

Note

The information in the *Select Reusable Files* window is taken from the job's reuse table (RUT) file. This information is lost if the RUT file is deleted.

In the *Select Reusable Files* window, do the following:

1 If this is your first time creating the output file, all the pages will be marked ✓ and ❗. Click **OK**.

For each page, a PS file is created and embedded in the output file.

2 If this is not your first time creating an output file for this job and the previous job exists on the press, leave the marks ✓ for the pages that you want embedded in the output JLYT file. These probably are the result of changes in the previous versions of the job residing on the press.

You can use **Select All**, **Select None**, and **Select Modified** to mark/unmark the pages in the Create column.

The job that YTD sends to the press contains the pages modified and created, and reuses (prints) the unmodified pages from the previous job. The previous job must exist on the press in order to reuse part or all of its reusable elements. The new job will appear on the press in addition to the previous job that is already there. Some elements are used by both jobs.

Note

After the next job is loaded to the press, it points at the required reusable elements from the previous job. These elements will not be deleted, even if you delete the original job from the press.

3 Click **OK** to start the Create Job process.

If you selected the **Printer** radio button in the *Create Job* window, an output JLYT or PPML file is now created and sent to the press you selected.

If you selected the **File** radio button in the *Create Job* window, an output JLYT or PPML file is created and saved in the Macintosh destination folder that you chose. Additionally, the job's RUT file is created/updated. The RUT file contains information on the job's reusable elements.

The output file creation may take some time.

Note

Besides the main QuarkXPress document, two additional files are created: the RUT file and the YTSE document file (if spread elements were added). If you archive your files, these three files should be archived to the same folder.

Output file structure

This section describes the organization of the YTD output file structure.

Project folder

Output files of project layouts are placed in a project folder created by the YTD software named "Project Name f."

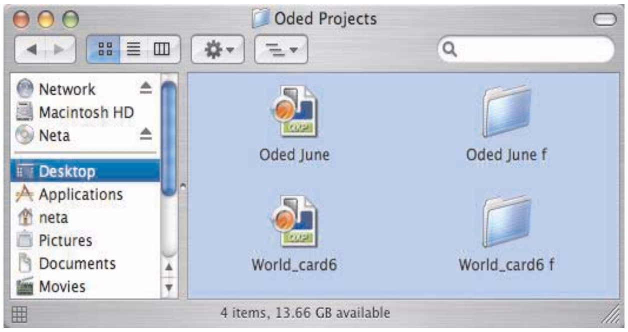


Figure 6-10. Folder for the project output files

Output files

If an output JLYT, PPML, or PDF file was created, it will be placed in the project folder along with the RUT file and a YTSE file (if it was created).

In the example shown in Figure 6-11, the project has two layouts, one with a PPML output file, and one with a JLYT file, both have a RUT file. Only one layout has a YTSE file. A PDF file was exported for both.

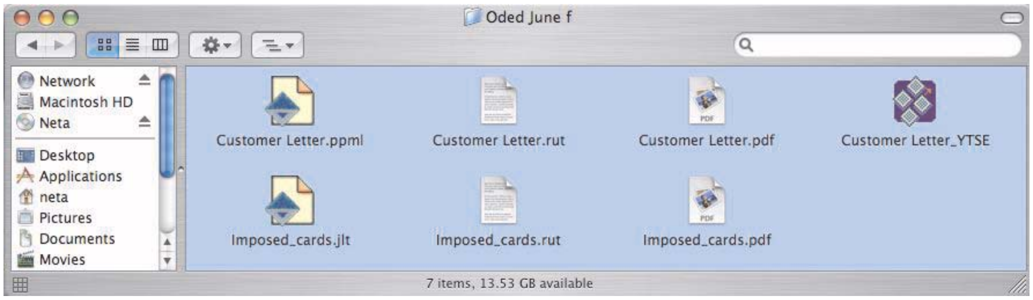


Figure 6-11. Project folder with output files

Exporting as a PDF file

YTD can output personalized or non-personalized jobs as PDF files. The jobs can be output as a single file or as several individual files, with or without imposition applied, or a combination of these two options. Including imposition produces a low-resolution in the PDF output and is useful for soft-proofing.

Note To use the **Export as PDF** feature, Xpress Tags Filter and PDF Filter extensions must be enabled (both are included in the original QuarkXPress extension set).

The PDF output file will be generated in CMYK only.

Exporting a PDF file

To output a PDF file, use the following steps:

1 In the QuarkXPress menu bar, click **Yours Truly** and then **Export as PDF**. The *Export PDF* window opens.

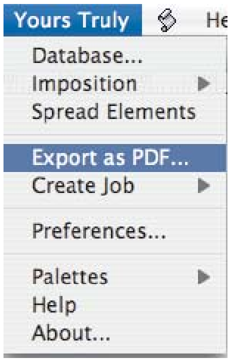
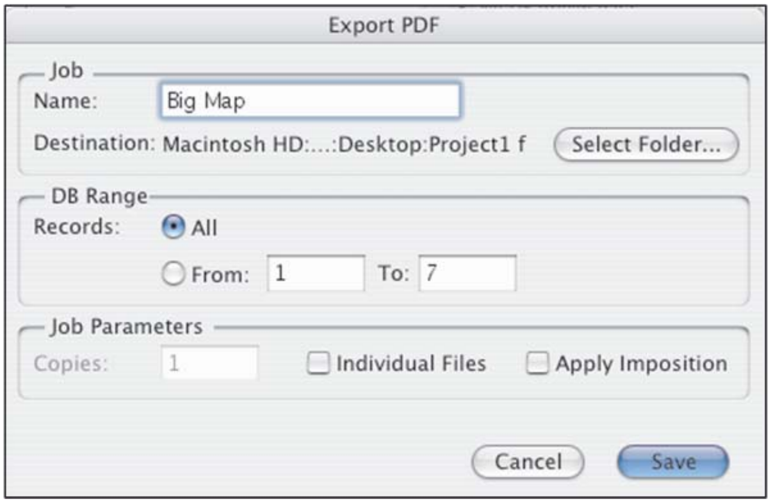


Figure 6-12. ***Yours Truly* - *Export as PDF* menu**

2 In the **Job** pane, *Name* field, the QuarkXPress layout name appears as a default name. Type a name for the output PDF file, if you wish to change it.

3 Click **Select Folder** to select a destination for the file. The default destination folder is the project name f folder.

4 For a personalization job, in the *Export PDF* window, in the **DB Range** pane, select **All** to export all records, or **From** and type the desired range of records.



Export PDF

Job
Name: Big Map
Destination: Macintosh HD:...:Desktop:Project1 f

DB Range
Records: ● All
○ From: 1 To: 7

Job Parameters
Copies: 1 □ Individual Files □ Apply Imposition

Figure 6-13. ***Export PDF*** **window**

5 In the Job Parameters pane, check the boxes as required:

Individual Files	Apply Imposition	Result
□	□	Creates a single multipage PDF file. Each page contains a single data record with no imposition.
☑	□	Creates a separate PDF file for each data record. The files do not include imposition
□	☑	Creates a single multipage PDF file. Each page contains data of one or more records based on the defined imposition.
☑	☑	Creates a separate PDF file for each data record. The files include imposition. (2-up imposition produces 2 records in one file).

6 Click **Save**. A PDF file is created according to the options selected.

Mounting the press shared volume

After a job is prepared, its output JLYT, PPML, or PDF file can be saved as a file on the Macintosh and later sent to the HP Indigo press for printing.

A JLYT or PPML output file can also be printed to a printer; that is, sent directly to the HP Indigo press.

In order to send the output JLYT, PPML, or PDF files that were created on the Macintosh to be printed, the HP Indigo press shared volume must be mounted and should appear on the Macintosh desktop.

Macintosh operating system 10

If it is not on the desktop, do the following:

1 Click **Go,** and select **Network**. The *Network* window opens.

2 Browse and select the name of the HP Indigo press.

3 Connect as Guest.

An icon representing the HP Indigo press shared volume appears on the desktop.

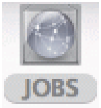


Figure 6-14. HP Indigo press shared volume icon

When you send an output personalized JLYT or PPML file to the HP Indigo press, you may need to make sure that personalization assets are present on the press. You can see a list of personalization assets necessary for the job when you create the output file, as described in this chapter.

Macintosh operating system 9.x

If it is not on the desktop, do the following:

1 Open the Chooser and click the **AppleShare** icon.

2 In the Select a File Server pane, select the HP Indigo press.

3 Click **OK**.

4 Connect as Guest.

An icon representing the HP Indigo press shared volume appears on the desktop.

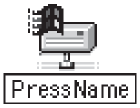


Figure 6-15. HP Indigo press shared volume icon

When you send the output JLYT or PPML file to the HP Indigo press, you may need to make sure that personalization assets are present on the press. You can see a list of personalization assets necessary for the job when you create the output file, as described in this chapter.

Sending the output JLYT or PPML file to the press

If you selected the **File** radio button in the *Create Job* window, an output JLYT or PPML file is created and saved in the destination folder you chose.

You need to send the output JLYT or PPML file to the press to be imported either manually or automatically. There are three options in how to manage the output JLYT or PPML file and its assets:

- Job with no personalization
- Output created as a Job
- Output created as a Template

These different options are described below.

Job with no personalization assets

If the job does not contain any personalization channels, place the JLYT or PPML output file:

- For the HP Indigo press 5000: `S:\JOBS\4COLORS` folder, where `S` is the drive letter.
- For all other HP Indigo presses: `S:\JOBS\PRESS\INPUT\LAN` folder, where `S` is the drive letter.

Output created as a Job

If the output JLYT or PPML file is created as a Job:

1. Make sure that the personalization assets listed in the *Assets* window are available to the press:
   - When necessary for JLYT or PPML files, download the fonts listed in the *Assets* window to the press. See "Downloading a font from the Macintosh operating system 9.x" on page 153 or “Downloading a font form the Macintosh operating system 10.x” on page 145 and "Creating a font on the press" on page 158.
   - If needed, make the images listed in the *Assets* window available to the press. Place the image files in the folder defined in the Images folder pane of the *Create Job* window (Figure 6-3.)
2. Place the output JLYT or PPML file:
   - For the HP Indigo press 5000: `S:\JOBS\4COLORS` folder, where `S` is the drive letter.
   - For all other HP Indigo presses: `S:\JOBS\PRESS\INPUT\LAN` folder, where `S` is the drive letter.

Output created as template

If the output JLYT nor PPML file is created as a Template:

1. Make sure that the personalization assets listed in the *Assets* window are available to the press:
   - When necessary for JLYT or PPML files, download the fonts listed in the *Assets* window to the press. See "Downloading a font from the Macintosh operating system 9.x" on page 153, or "Downloading a font from the Macintosh operating system 10.x" on page 156 and "Creating a font on the press" on page 158.

- If needed, make the images listed in the *Assets* window available to the press. Place the image files in the folder defined in the Images folder pane of the *Create Job* window (Figure 6-3.)

Automatic workflow:

2 For the HP Indigo press 5000, if the output JLYT or PPML file and the DB files have the same name (for example, *customers.jlt and customers.csv, or customers.txt*), perform an automatic import to the HP Indigo press 5000 press:

a Place the output JLYT or PPML file in the `S:\JOBS\4COLORS\TEMPLATE` folder.

b Place the output DB report file in the `S:\JOBS\4COLORS` folder.

or

a Place the output DB report file in the `S:\JOBS\4COLORS\DB` folder.

b Place the output JLYT or PPML file in the `S:JOBS\4COLORS` folder.

3 For all other HP Indigo presses, if the output JLYT file and the DB files have the same name (for example, *customers.jlt and customers.csv, or customers.txt*), perform an automatic import to the press:

a Place the output JLYT file in the `S:\JOBS\PRESS\TEMPLATE` folder.

b Place the DB report file in the `S:\JOBS\PRESS\INPUT\LAN` folder.

or

a Place the DB report file in the `S:\JOBS\PRESS\DB` folder.

b Place the output JLYT file in the `S:\JOBS\PRESS\INPUT\LAN` folder.

Manual workflow:

4 For the HP Indigo press 5000, if the output JLYT or PPML file and the DB files do not have the same name, you must import the files manually to the press:

a Place the DB report file in the `S:\JOBS\4COLORS\DB` folder.

b Place the output JLYT or PPML file in the `S:\JOBS\4COLORS\TEMPLATE` folder.

c In the press Job Manager, select **Import**. The *Import* window opens.

d Browse and select the output JLYT or PPML file.

e In the *DB File Preview* window, browse and select the DB file.

5 For all other HP Indigo presses, if the output JLYT file and the DB files do not have the same name, you must import the files manually to the press:

a Place the DB report file in the `S:\JOBS\PRESS\DB` folder.

b Place the output JLYT file in the `S:\JOBS\PRESS\TEMPLATE` folder.

c In the press Job Manager, select **Import**. The *Import* window opens.

d Browse and select the output JLYT file.

e In the *DB File Preview* window, browse and select the DB file.

Sending the PDF output files to the press

If you selected **Export as PDF...** from the **YTD** menu, PDF output file(s) are created and saved at the destination folder you chose.

These files need to be sent to the press where they will be RIPped and later printed. The PDF files should be placed in the *S:\JOBS\RIP\INPUT\4COLORS* hot folder, or other RIP hot folder according to the job needs. For the HP Indigo press 5000 place the PDF files in S:\JOBS\4COLORS folder.

7 Changing YTD preferences

This chapter contains the following topics:

- Overview
- Changing global parameters
- Changing paper parameters
- Changing marks parameters
- Changing image parameters
- Changing output parameters

Overview

YTD installs with a set of default values for various YTD system parameters, such as paper size, margins, and crop marks. This section describes how to change these defaults, if necessary.

1 Click **Yours Truly** and **Preferences**.

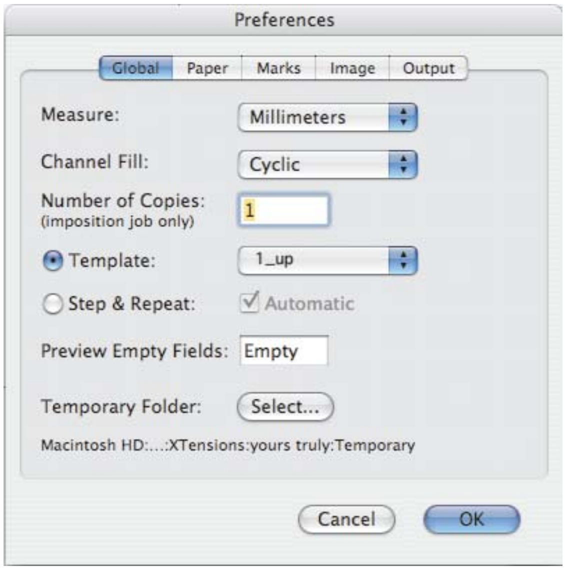
The *Preferences* window appears.

2 Select the relevant tab(s) and change the values of the relevant fields, as described below.

3 Click **OK**.

Changing global parameters

Click the **Global** tab.



Measure: Millimeters

Channel Fill: Cyclic

Number of Copies: 1
(imposition job only)

● Template: 1_up

○ Step & Repeat: ☑ Automatic

Preview Empty Fields: Empty

Temporary Folder:

Macintosh HD:...:XTensions:yours truly:Temporary

Figure 7-1. Global parameters tab

The following parameters can be changed:

- **Measure** — In the **Measure** drop-down menu, select the measurement unit (**Millimeters** or **Inches**).
- **Channel Fill** — Printing a copy of a personalization job prints all database records one time. If more than one copy of the job is printed, the Channel Fill field determines what is printed in the channel cycles after the first copy.

  From the **Channel Fill** drop-down menu, select the relevant option:

  - **Cyclic**: All the channel cycles of the first copy are repeatedly printed in the channels of the subsequent copies.
  - **Empty**: Nothing is printed in the channels of the subsequent copies.
  - **Extend**: All the channel cycles of the last sheet of the first copy are repeatedly printed in the channels of the subsequent copies.

In the following example of the Channel Fill options, the job consists of a one-page QuarkXPress document with one personalization channel of 12 cycles. Each printed copy of the job produces two sheets with six cycles per sheet. The example below shows two printed copies of the job.

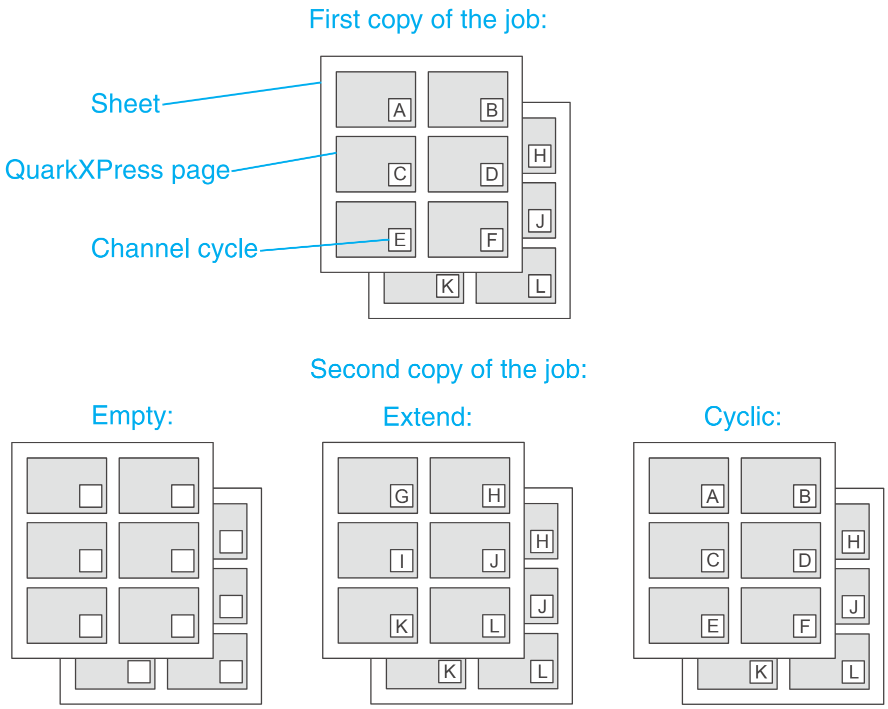
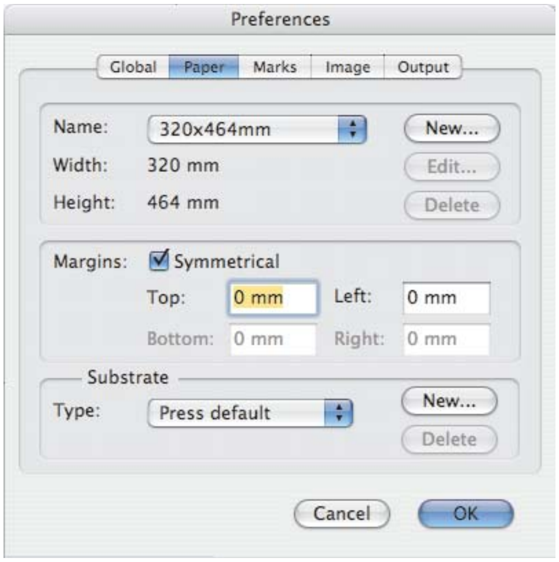


Figure 7-2. Examples of channel fill

- **Number of Copies** — To change the default number of copies to print of an imposition only job (see *Create Job* window in "Creating an output file" on page 121), type the number in the **Number of Copies** field.
- **Template/Step & Repeat** — To change the default imposition type (see "Layout tab" on page 89), select one of the following:
  - **Template** radio button and a template from the drop-down menu.
  - **Step & Repeat** radio button and the **Automatic** check box.
- **Preview Empty Fields** — To define the appearance of empty fields, either leave the default value as "Empty," type another value, or remove all characters.
- **Temporary Folder** — The file path that appears at the bottom of the window leads to the folder in which YTD stores and deletes temporary files. It is recommended that you do not change the location of this folder.

Changing paper parameters

Click the **Paper** tab.



Name: 320x464mm
Width: 320 mm
Height: 464 mm
Margins: ☑ Symmetrical
Top: 0 mm Left: 0 mm
Bottom: 0 mm Right: 0 mm
Substrate
Type: Press default

Figure 7-3. Paper tab

When you impose a job, you verify or select an appropriate paper for the job (see "Imposing a job using JLYT imposition" on page 87, and "Imposing a job using PPML imposition" on page 103). The **Paper** tab allows you to add, edit, and delete paper definitions. You also determine the default paper and margin values for imposition.

To add a new paper definition, do the following:

1 Click **New**.

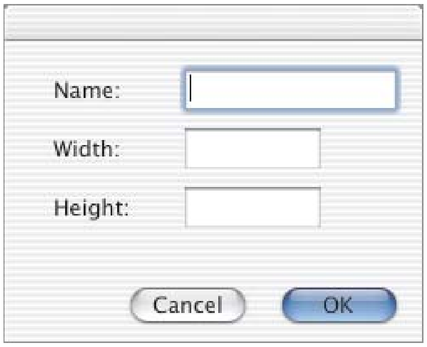


Figure 7-4. *New paper* window

2 In the window that appears, type paper definition values for **Name**, **Width**, and **Height**.

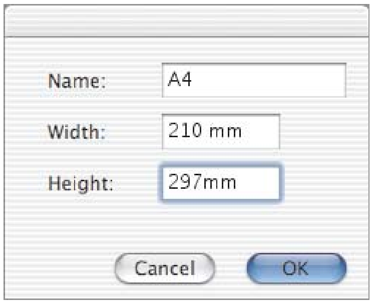
3 Click **OK**.

To edit an existing paper definition, do the following:

1 In the **Name** drop-down menu, select the paper definition.

Note Only paper definitions you add (as described above) can be edited. Paper definitions supplied with YTD cannot be edited.

2 Click **Edit**.


Name: A4

Width: 210 mm

Height: 297mm

Figure 7-5. ***Edit paper*** **window**

3 In the window that appears, change the paper definition values for **Name**, **Width**, and/or **Height**.

4 Click **OK**.

To delete an existing paper definition, select the paper definition, click **Delete** and then **OK**.

The paper that appears in the **Paper** tab when you close the *Preferences* window is the default paper. If you want a different paper to be the default, select it before you close the *Preferences* window.

To change the default margin values for the imposition **Paper** tab (see "Paper tab" on page 88 for descriptions of these fields), do the following:

1 Select or clear the **Symmetrical** check box.

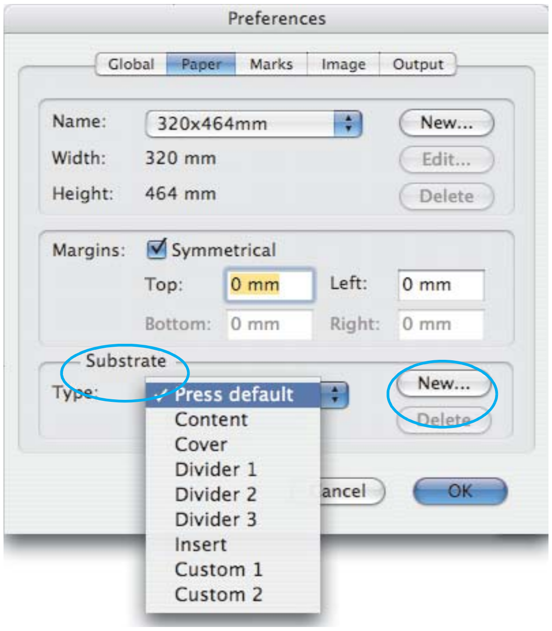
2 Change the margin values in the **Top**/**Bottom** and **Left**/**Right** fields.

Substrate names can be defined for a job sheet in YTD *Imposition* window, **Substrate** tab. The predefined substrate list consists of reserved words (Press Default, Cover, Content, Insert, Divider, Custom). The same reserved words also exist at the HP Indigo press 5000 as intended names.

Note Substrate definitions are ignored when an output file which contains substrate definitions is sent to any press (other than HP Indigo press 5000) that does not support substrate definition,

You can add additional substrates names to the list in the *Preferences* window on the **Paper** tab.

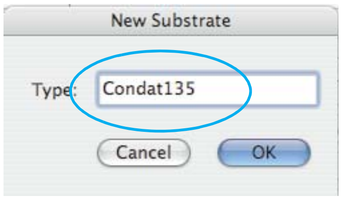
3 In the *Substrate* pane click **New**



Name: 320x464mm
Width: 320 mm
Height: 464 mm
Margins: ☑ Symmetrical
Top: 0 mm
Left: 0 mm
Bottom: 0 mm
Right: 0 mm
Type: Press default

Figure 7-6. Adding a new substrate

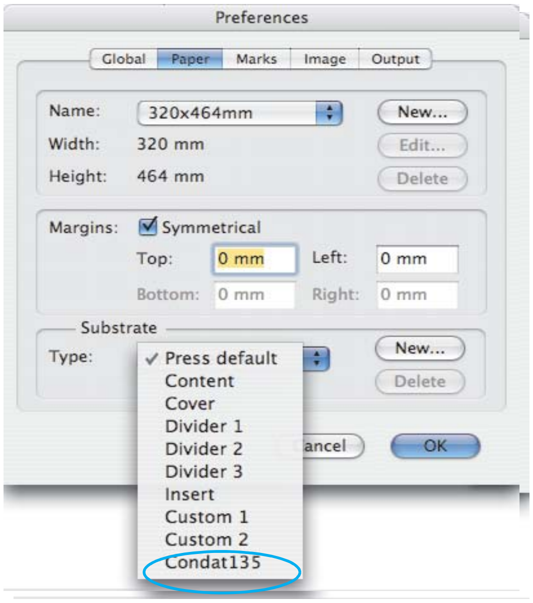
4 In the *New Substrate* window that opens type a new name and click **OK**.



Type: Condat135

Figure 7-7. Typing a new substrate name

The new name is appended to the existing list.

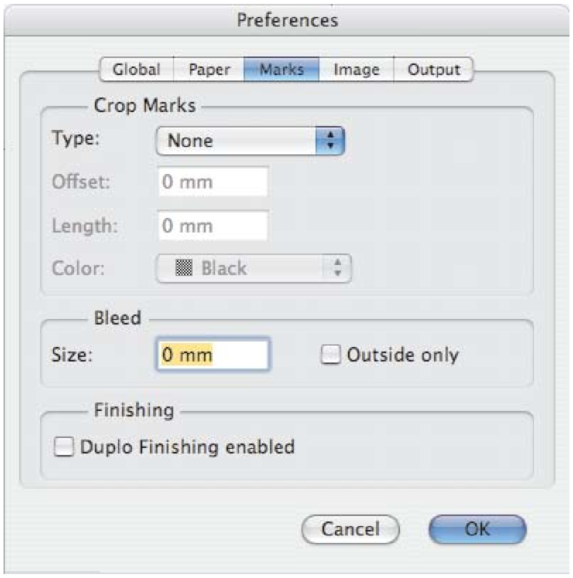


Name: 320x464mm
Width: 320 mm
Height: 464 mm
Margins: ☑ Symmetrical
Top: 0 mm Left: 0 mm
Bottom: 0 mm Right: 0 mm
Substrate
Type: Press default

Figure 7-8. ***Substrates* window appended**

Changing marks parameters

Click the **Marks** tab.



Crop Marks
Type: None
Offset: 0 mm
Length: 0 mm
Color: ▨ Black
Bleed
Size: 0 mm ☐ Outside only
Finishing
☐ Duplo Finishing enabled

Figure 7-9. Marks parameters tab

To change the default crop mark values for the imposition **Marks** tab, select or type the relevant values for the relevant fields:

- **Type** drop-down menu
- **Offset**
- **Length**

- **Color** drop-down menu
- **Bleed** (size)
- **Outside only** check box.
- **Duplo Finishing enabled** check box

See "Marks tab" on page 92 for descriptions of these fields.

Changing image parameters

Click the **Image** tab.

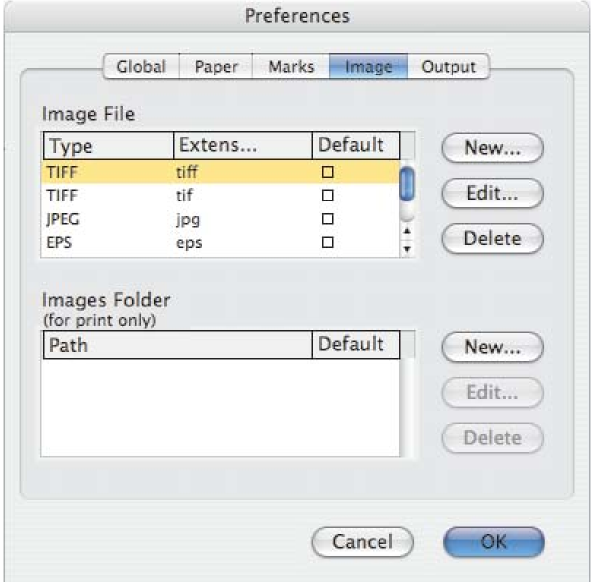


Image File

Type	Extens...	Default
TIFF	tiff	□
TIFF	tif	□
JPEG	jpg	□
EPS	eps	□

Images Folder
(for print only)

Path	Default

Figure 7-10. Image tab

Image file

When you define a personalization image channel (see "Defining an image channel" on page 34), you must verify the correct file type and extension of the image files that will be printed in the channel. Image types that are allowed and recognized by YTD are TIFF, JPEG, EPS, PDF, JLYT, and PS. Each file type can have one or more file extensions, or no file extension, associated with it. The image file name, including its extension, must be identical to its corresponding file name (and extension) in the DB report file. If the filename contains an extension and its filename in the DB report file does not contain an extension, you can add the extension to the list of valid extensions for selection.

The Image File pane allows you to add, edit, and delete file extensions that are associated with the YTD-recognized image types.

To add a new extension to an image type, do the following:

1 Click **New**. The *New Image Extension* window appears.

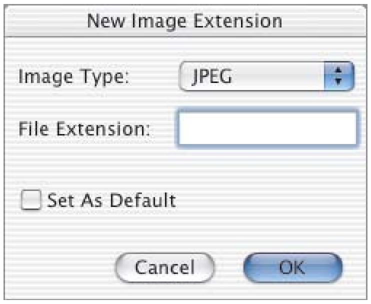


Image Type: JPEG

File Extension:

□ Set As Default

Figure 7-11. *New Image Extension* window

2 Select the file type from the **Image Type** drop-down menu.

3 In the **File Extension** field, type the new file extension.

4 Select the **Set As Default** check box if you want this file extension to be the default extension when selecting the image type for the fast image channel.

5 Click **OK**.

To edit an existing file extension, do the following:

1 Select the file extension.

2 Click **Edit**. The *Edit Image Extension* window appears.

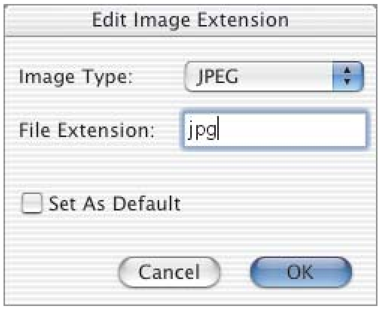


Image Type: JPEG

File Extension: jpg

□ Set As Default

Figure 7-12. *Edit Image Extension* window

3 Type the new file extension in the **File Extension** field.

4 Select the **Set As Default** check box if you want this file extension to be the default extension when selecting the image type for the fast image channel.

5 Click **OK**.

To delete an existing file extension, select the file extension and click **Delete**.

Images folder

When you create an output JLYT or PPML file (see "Creating an output file" on page 121), you choose the path of the variable images files that are accessed by the job on the press. If the job's variable images files are not at the press default path, you can either type the path or select a previously saved path from a drop-down menu list.

The Images Folder pane allows you to add, edit, and delete paths.

To add a path, do the following:

1 Click **New**. The *New Images Path* window appears.

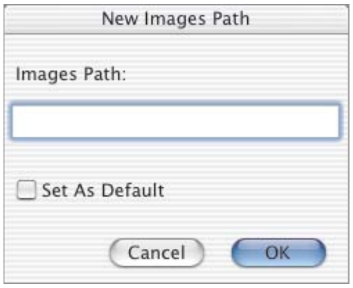


Figure 7-13. *New Images Path* window

2 Type the path in the **Images Path** field as either a complete path or a sub-folder path to be appended to the press' default path.

Note A complete path includes a drive letter and colon. For example: S:\jobs\images
An appended path should not have a drive letter. For example: images1\pics

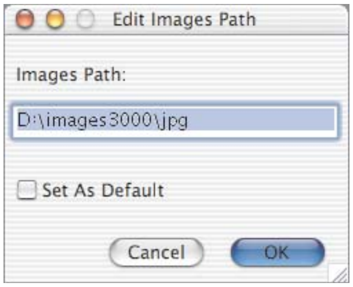
3 Select the **Set As Default** check box if you want this path to be the default when choosing a path for the variable images files.

4 Click **OK**.

To edit an existing path, do the following:

1 Select the path.

2 Click **Edit**. The *Edit Images Path* window appears.



D:\images3000\jpg

Figure 7-14. *Edit Images Path* window

3 Edit the path in the **Images Path** field.

4 Select the **Set As Default** check box if you want this path to be the default when choosing a path for the variable images files.

5 Click **OK**.

To delete an existing path, select the path and click **Delete**.

Changing output parameters

Click the **Output** tab.

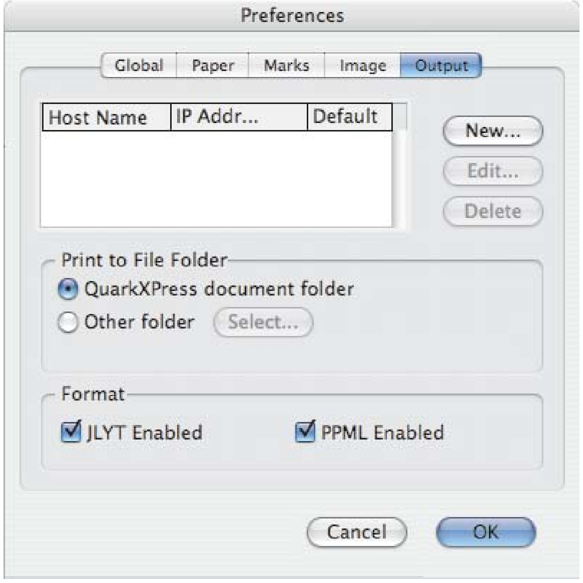


Figure 7-15. Output tab

When you create an output JLYT or PPML file, you can send the file directly to an HP Indigo press or save it on the Macintosh. If you send it directly to an HP Indigo press, the press must be defined and configured.

The **Output** tab allows you to add, edit, and delete HP Indigo press (printer) definitions.

Printer definition

To add a printer definition, do the following:

1 Click **New**. The *New Printer* window appears.

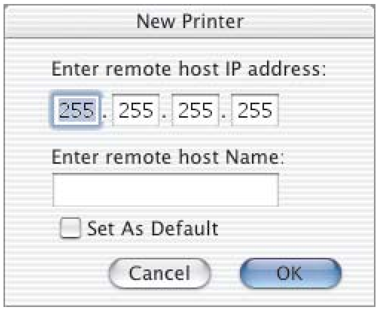


Figure 7-16. ***New Printer*** **window**

2 In the **IP address** field, type the printer's IP address.

3 In the **Name** field, type a printer name.

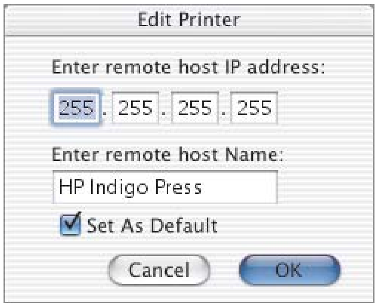
4 Select the **Set As Default** check box if you want this printer to be the default when selecting the **Printer** radio button in the *Create Job* window.

5 Click **OK**.

To edit an existing printer definition, do the following:

1 Select the printer line.

2 Click **Edit**. The *Edit Printer* window appears.



Edit Printer

Enter remote host IP address:

255 . 255 . 255 . 255

Enter remote host Name:

HP Indigo Press

☑ Set As Default

Figure 7-17. ***Edit Printer*** **window**

3 Edit the printer's **IP address** and **Name**.

4 Select the **Set As Default** check box if you want this printer to be the default when selecting the **Printer** radio button in the *Create Job* window.

5 Click **OK**.

To delete an existing printer, select the printer and click **Delete**.

Print-to-file folder definition

The Print-to-file Folder pane is used to define the location into which the output JLYT or PPML file is placed when it is printed to a file. The default file location is the folder in which the QuarkXPress project (document) is placed.

To modify a Print-to-file Folder definition in a new folder created by YTD software, named Project_Name f, do the following:

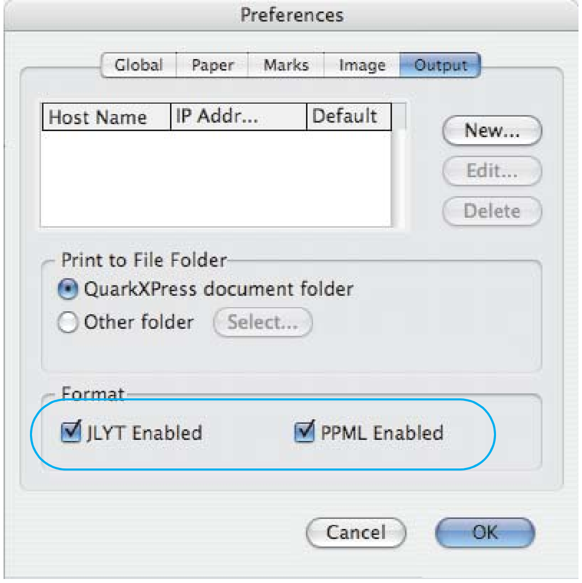
1 Click **Other Folder** radio button. The **Select** button becomes enabled.

2 Click **Select**, and browse to the desired destination folder.

Note During the job creation process, you can use the **Select Folder** button in the *Create Job* window to select a destination folder (See "Creating an output file" on page 121.) This destination can be different from the destination defined in the **Printer** tab**,** in the *YTD Preferences* window.

Format definition

The Format pane is used to define the output format as **JLYT enabled**, **PPML enabled** or both.



Print to File Folder

● QuarkXPress document folder

○ Other folder

Format

☑ JLYT Enabled ☑ PPML Enabled

Figure 7-18. Defining output format in *Preferences* window

- When JLYT is enabled, the YTD classic imposition is available, and a JLYT output file is generated when you create a job.
- When PPML is enabled, PPML imposition becomes available in the Imposition menu, and a PPML output file is generated when you create a job.
- When both JLYT and PPML are enabled, both imposition methods are available, and both types of output files can be generated when you create a job. Also, you can select JLYT imposition and create a PPML output file. However, not all JLYT features are supported when creating a PPML output, and not all PPML features are supported when creating a JLYT output.

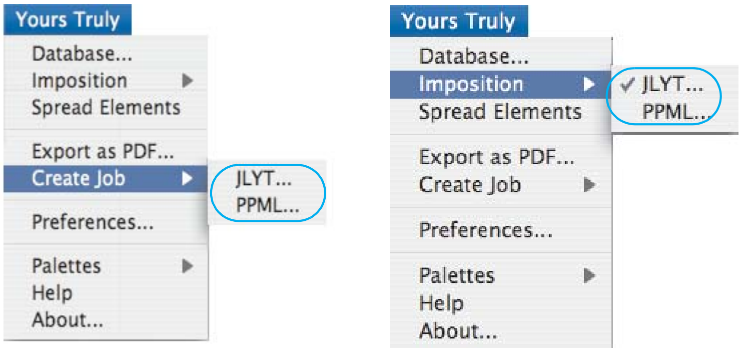


Figure 7-19. PPML in *Create Jobs* and *Imposition* windows

8 SNAP fonts

This chapter contains the following topic:

- Overview
- Checking the press font list
- Downloading a font from the Macintosh operating system 9.x
- Downloading a font from the Macintosh operating system 10.x
- Using the FontConverter
- Creating a font on the press

Overview

When creating the output JLYT or PPML file for a personalization job, the *Assets* window (Figure 6-5., page 125) and personalization assets report may contain a list of specific fonts and font sizes that need to be available to the press.

At the press, these fonts must undergo a transformation from PostScript to SNAP. This is referred to as “creating” the font.

Creating a font is a one-time procedure for the specific font and size. Afterwards, the specific font size can be used repeatedly when needed by personalization jobs.

When font is listed on the personalization assets report, one of the three following possibilities exist:

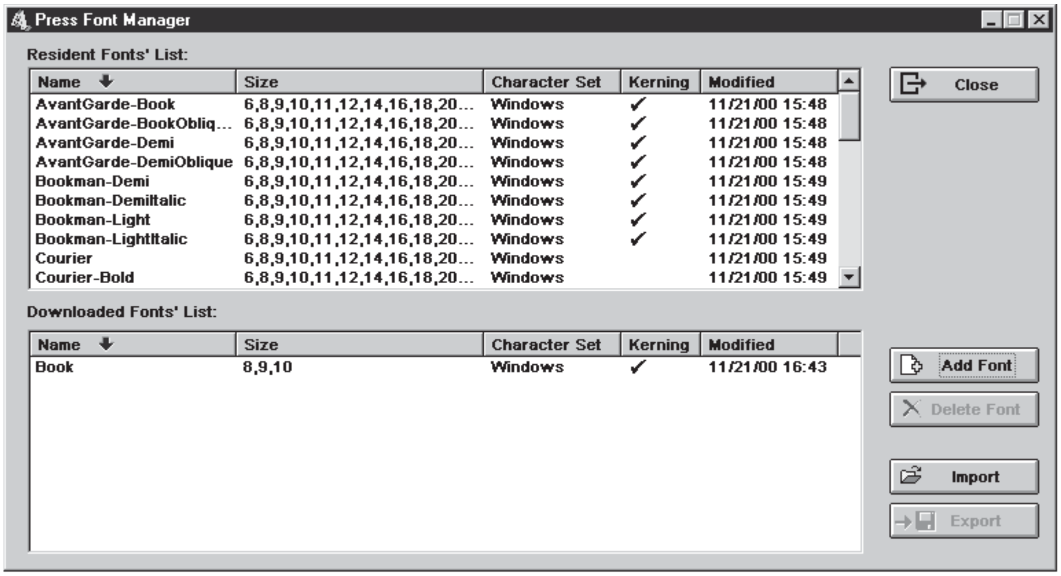
- The font and size exist and were created on the press. In this case, you do not need to do anything about the font.
- The font exists on the press but has not yet been created at all or has not yet been created for the specific sizes needed. In this case, perform the procedure described in "Creating a font on the press" on page 158.
- The font does not exist on the press. In this case, you need to download the font and create it on the press. Perform the procedure described in "Downloading a font from the Macintosh operating system 9.x" on page 153, or “Downloading a font from the Macintosh operating system 10.x” on page 145, followed by the procedure described in "Creating a font on the press" on page 158.

The following procedures describe how to check if a particular font and font size exist on the press, how to download the font from the Macintosh, and how to create it on the press.

Checking the press font list

To check if a particular font exists on the press, click **Options** and **Font Manager** at the press computer.

The *Press Font Manager* window appears.



Press Font Manager

Resident Fonts' List:

Name	Size	Character Set	Kerning	Modified
AvantGarde-Book	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-BookObliq...	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-Demi	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-DemiOblique	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
Bookman-Demi	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-DemiItalic	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-Light	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-LightItalic	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Courier	6,8,9,10,11,12,14,16,18,20...	Windows		11/21/00 15:49
Courier-Bold	6,8,9,10,11,12,14,16,18,20...	Windows		11/21/00 15:49

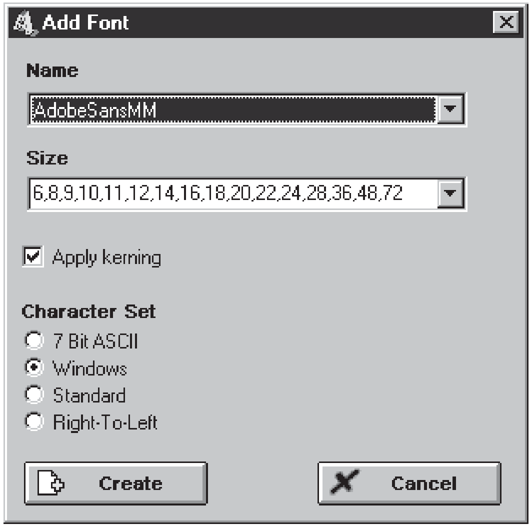
Downloaded Fonts' List:

Name	Size	Character Set	Kerning	Modified
Book	8,9,10	Windows	✓	11/21/00 16:43

Figure 8-1. ***Press Font Manager*** **window**

The Resident Fonts' List and Downloaded Fonts' List contain all the fonts that exist were created on the press.

To find a PostScript font that exists on the press but was not created, click **Add Font**.



Name
AdobeSansMM

Size
6,8,9,10,11,12,14,16,18,20,22,24,28,36,48,72

☑ Apply kerning

Character Set
○ 7 Bit ASCII
● Windows
○ Standard
○ Right-To-Left

Figure 8-2. *Add Font* window

In the *Add Font* window that appears, search for the font in the **Name** drop-down list.

If the font is not in the Name drop-down list, it does not exist on the press and must be downloaded from the Macintosh. See "Downloading a font from the Macintosh operating system 9.x" or "Downloading a font from the Macintosh operating system 10.x", below.

If the font is in the Name drop-down list, it must be created as a SNAP font. See "Creating a font on the press" on page 158.

Downloading a font from the Macintosh operating system 9.x

If the font does not exist on the list, and you need to download it from the Macintosh. For HP Indigo press 5000, start "At the Macintosh computer" on page 155. For all other HP Indigo presses, start below.

At the press computer

1 Click **Start**, **programs**, **hp indigo RIP** (for all presses except the HP Indigo press 5000), and **RIP GUI**.

The hp indigo RIP toolbar appears.

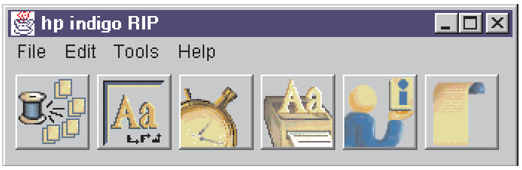


Figure 8-3. hp indigo RIP toolbar

2 From the hp indigo RIP toolbar, click the **Font Downloader** button or click **Tools** and **Font Downloader**.

The *Font Downloader* window appears.

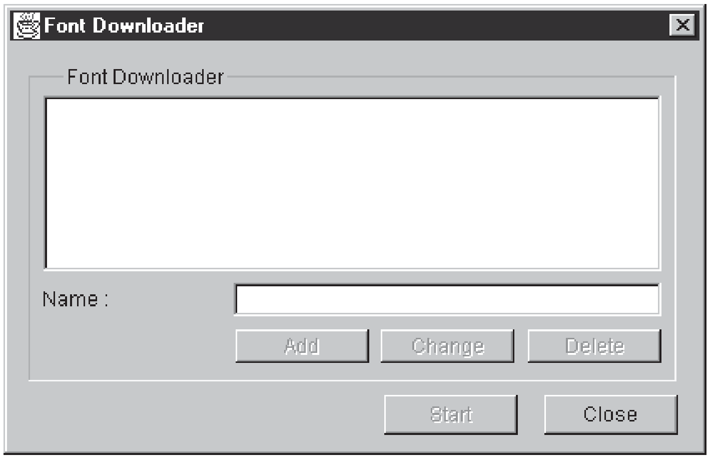


Figure 8-4. ***Font Downloader*** **window**

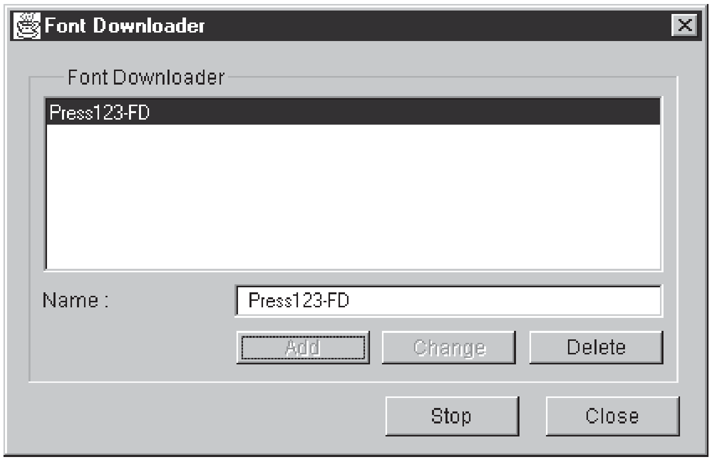
3 If a name appears in the Font Downloader box, record the name for further reference. If there is no name in the Font Downloader box, type a font downloader name in the **Name** field. For example: `Press123-FD`

Record the font downloader name for further reference.

4 Click **Add**.

Note A font downloader name is sometimes referred to as a Printer Access Protocol.

The font downloader name appears in the Font Downloader box.



Press123-FD

Name : Press123-FD

Figure 8-5. Adding a font

5 Click **Close**.

At the Macintosh computer

1 Open the Chooser and select the **LaserWriter** icon.

2 **For HP Indigo press 5000**: In the PostScript Printer list, select the HP Indigo press 5000 name: “Press name HP RIP Fonts”

3 **For all other HP Indigo presses:** In the Select a PostScript Printer pane, select the font downloader name that you recorded earlier (see step 3 above) at the press computer.

4 Quit the Chooser.

5 Open the Adobe Font Downloader (version 5.05 or later) or a different font downloader that you use.

The following instructions refer to the Adobe Font Downloader.

6 Click **File** and **Download Fonts**.

The *Outline Fonts* window appears.

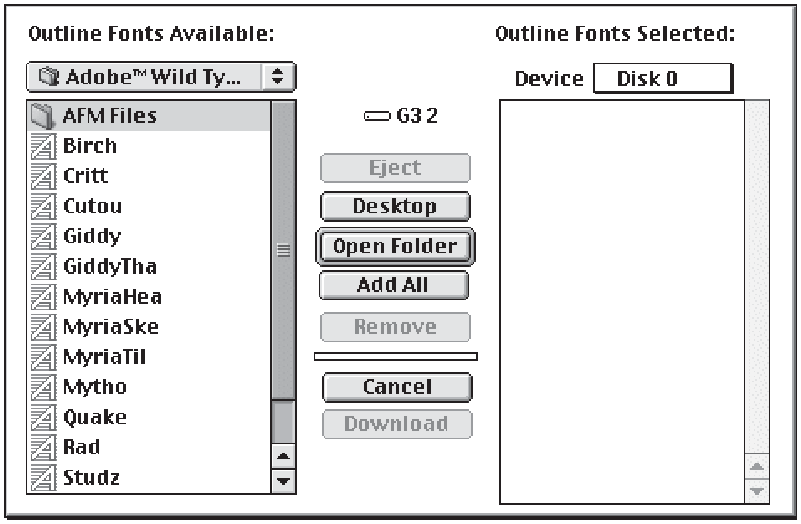


Figure 8-6. ***Outline Fonts*** **window**

7 In the **Device** drop-down menu, select **Disk 0**.

8 In the Outline Fonts Available list, select the font that you want to download to the press, and then click **Add**. You can repeat this for all the fonts you want to download.

The fonts that you selected appear in the **Outline Fonts Selected** list.

9 Click **Download**.

A status message window appears during the downloading.

10 When `The download was successful` message window appears, click **OK**.

The fonts have been successfully downloaded to the press.

11 Open the Chooser, select your default PostScript printer, and quit the Chooser.

You may now create the font. See "Creating a font on the press" on page 158.

Downloading a font from the Macintosh operating system 10.x

If the font does not exist on the press font list, and you need to download it from the Macintosh, do the following at the Macintosh:

1 Insert the YTD software CD-ROM into the drive.

2 Double-click the CD-ROM icon to display its contents.

3 Double-click the **MAC OS Utils** folder.

4 Copy the FontConverter utility from the YTD software CD-ROM to your Macintosh hard disk.

Using FontConverter

FontConverter is a utility that enables converting fonts from Mac OS X, which can be then be downloaded to the HP Indigo press RIP.

Supported format conversions by **FontConverter** include:

- Mac TrueType to PC TrueType
- Mac Type1 to PFB file
- OpenType to OpenType

The converted font file can be saved to the Mac hard disk, or to the HP Indigo press shared volume.

Note

OpenType format is supported by **HP High Performance RIP**, but not by the **hp indigo RIP**.

Font conversion from MAC OS 9.x may be performed using **Adobe Downloader 5.0.5** or **Apple Printer Utility 2.2**

In order to use the **FontConverter** use one of the following workflows:

1 Launch the **FontConverter** application.

a Double click the **FontConverter** application.

b In the *Choose Font Files* window that opens, browse to and select the font files that you want to convert.

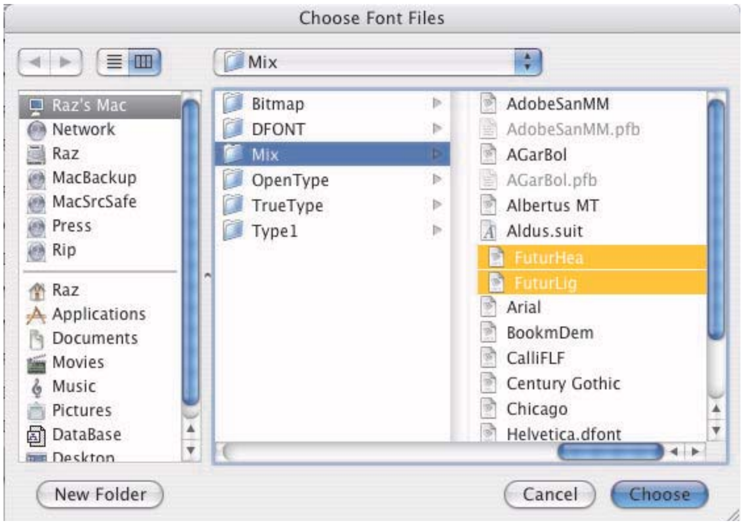


Figure 8-7. ***Choose Font Files*** **window**

Note You can choose multiple fonts to convert at one time.

c Click **Choose**.

d In the *Choose Destination Folder* window that opens, browse to and select the destination folder.

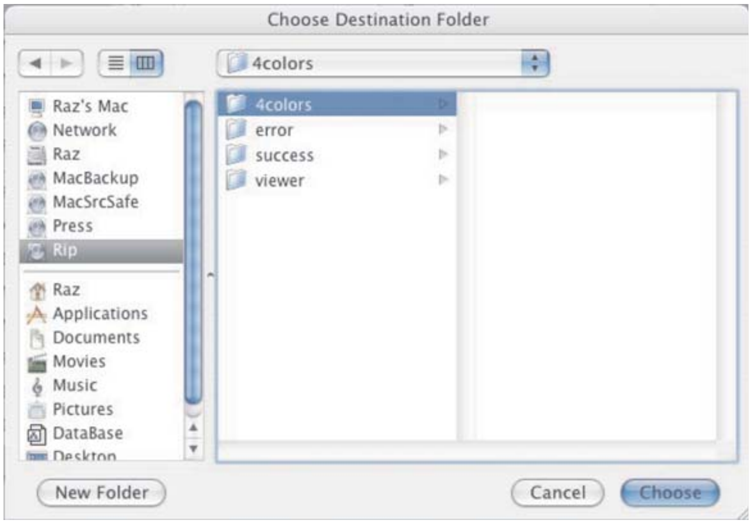


Figure 8-8. ***Choose Destination Folder*** **window**

e Click **Choose**.

A message will appear indicating the success or failure of the procedure.

OR

2 Drag and drop the fonts.

a Drag and drop the font files on the utility icon.

b In the *Choose Destination Folder* window that opens, browse to and select the destination folder

A message will appear indicating the success or failure of the procedure.

Downloading converted fonts to the press RIP

Once you have converted the fonts you can download them to the press RIP.

1 In the **HP Indigo press 5000** Production Manager, the converted font should be copied to the fonts hotfolder: `S:\JOBS\FONTS`.

In the **HP Indigo press** (other than the **HP Indigo press 5000**), The converted font file may be directed to the hp indigo RIP hot folder, `S:\JOBS\RIP\INPUT\4COLORS`, to be automatically imported to the RIP or drag and drop the fonts into the `S:\JOBS\RIP\INPUT\4COLORS` folder.

The font will be added automatically to the font list.

You may now create the font. See "Creating a font on the press" below.

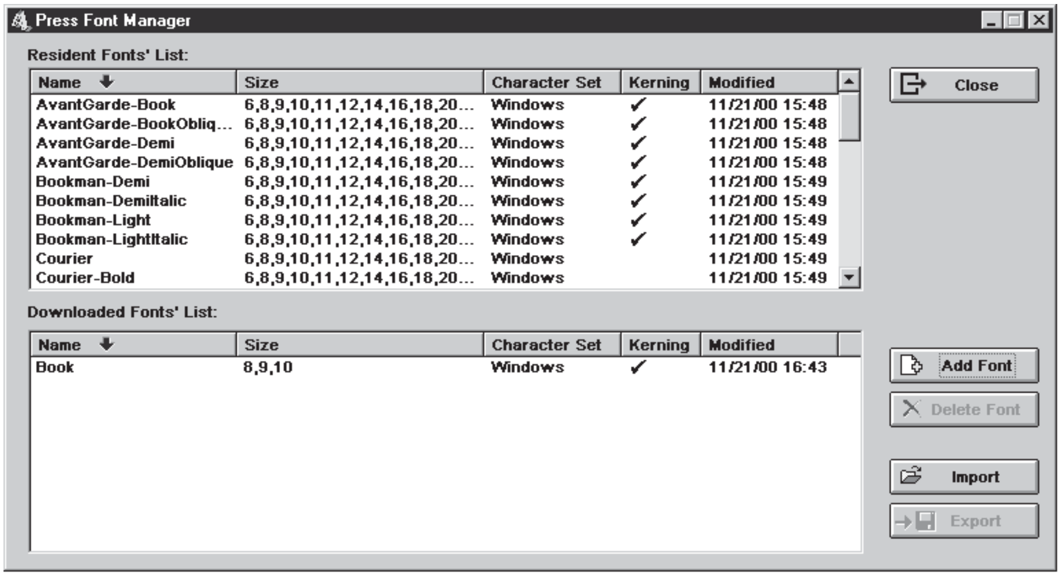
Creating a font on the press

Fonts that were downloaded from the Macintosh are created on the press. You can create the font using the Font Manager or during job import.

Creating the font using the font manager

1 Click **Options** and **Font Manager**.

The *Press Font Manager* window appears.



Resident Fonts' List:

Name	Size	Character Set	Kerning	Modified
AvantGarde-Book	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-BookObliq...	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-Demi	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-DemiOblique	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
Bookman-Demi	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-DemiItalic	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-Light	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-LightItalic	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Courier	6,8,9,10,11,12,14,16,18,20...	Windows		11/21/00 15:49
Courier-Bold	6,8,9,10,11,12,14,16,18,20...	Windows		11/21/00 15:49

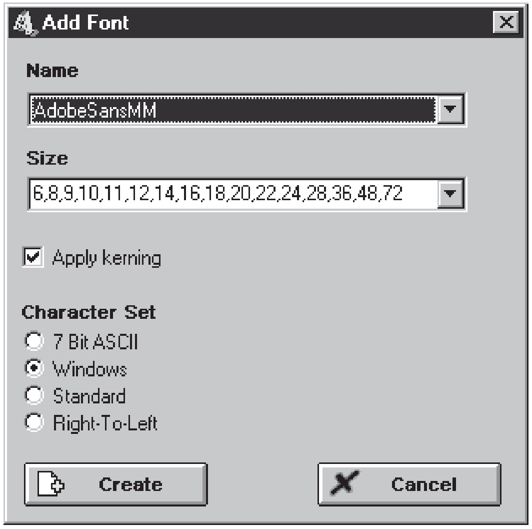
Downloaded Fonts' List:

Name	Size	Character Set	Kerning	Modified
Book	8,9,10	Windows	✓	11/21/00 16:43

Figure 8-9. ***Press Font Manager* window**

2 Click **Add Font**.

The *Add Font* window appears.



Name

AdobeSansMM

Size

6,8,9,10,11,12,14,16,18,20,22,24,28,36,48,72

☑ Apply kerning

Character Set

○ 7 Bit ASCII

● Windows

○ Standard

○ Right-To-Left

Figure 8-10. ***Add Font*** **window**

3 In the **Name** drop-down list, select the font that you want to create.

4 Erase the **Size** field contents and type the font sizes that you want for the font, separating the sizes with commas. Font sizes can be between 5 and 400 and must be whole numbers.

5 If the font you are creating is used in a bar code, clear the **Apply kerning** check box. Otherwise, leave it selected.

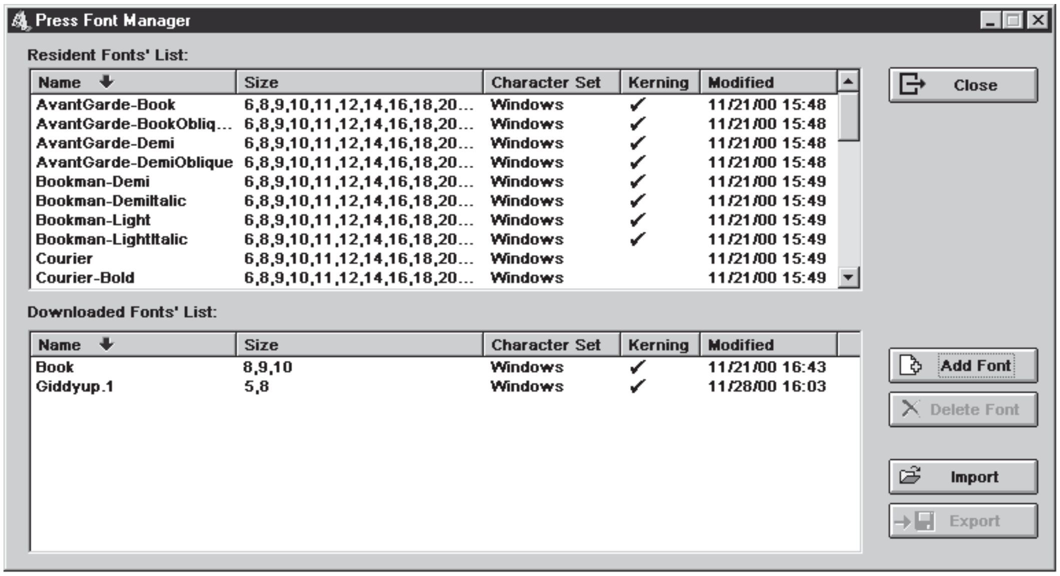
6 For the **Character Set** radio buttons, do the following:

- Select **Windows** if you want the default (for Roman fonts).
- Select **7 Bit ASCII** if the font is used for European diacritic characters.
- Select **Standard** if the font is used in bar code.
- Select **Right-To-Left** if the font is Hebrew or Arabic.

7 Click **Create**.

The press creates the selected font in all the chosen sizes.

After the font is created it appears in the Downloaded Fonts' List in the *Press Font Manager* window.



Press Font Manager

Resident Fonts' List:

Name	Size	Character Set	Kerning	Modified
AvantGarde-Book	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-BookObliq...	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-Demi	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
AvantGarde-DemiOblique	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:48
Bookman-Demi	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-DemiItalic	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-Light	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Bookman-LightItalic	6,8,9,10,11,12,14,16,18,20...	Windows	✓	11/21/00 15:49
Courier	6,8,9,10,11,12,14,16,18,20...	Windows		11/21/00 15:49
Courier-Bold	6,8,9,10,11,12,14,16,18,20...	Windows		11/21/00 15:49

Downloaded Fonts' List:

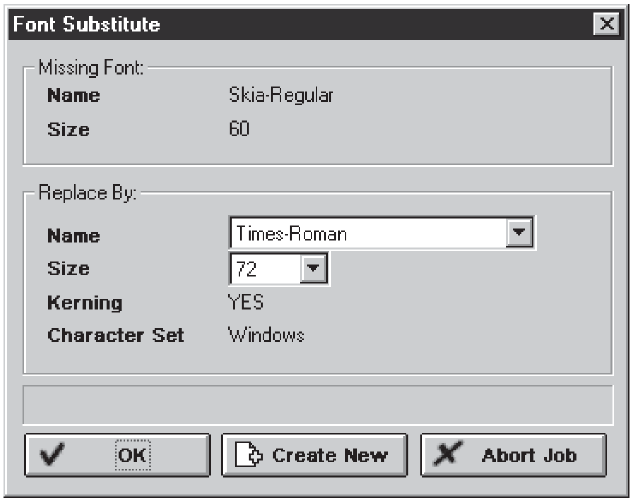
Name	Size	Character Set	Kerning	Modified
Book	8,9,10	Windows	✓	11/21/00 16:43
Giddyup.1	5,8	Windows	✓	11/28/00 16:03

Figure 8-11. Creating a font

You can repeat this procedure for each font you want to create.

Creating the font during job import

If you did not create the font before the job was imported, you will be asked by the press software (in the window shown below) to create the font, replace (substitute) the font with an existing one, or to abort the job.



Font Substitute

Missing Font:

Name Skia-Regular

Size 60

Replace By:

Name Times-Roman

Size 72

Kerning YES

Character Set Windows

Figure 8-12. Substituting a font

If you want to create the font during job import, do the following:

1 Click **Create New**.

The *Add Font* window appears. See Figure 8-10.

2 Proceed to the step after Figure 8-10. and continue from there.

9 DUPLO Finisher support

This chapter contains the following topic:

- Overview
- Enabling Duplo finishing

Overview

YTD 7 provides barcode support for two Duplo finishers: the Duplo DC 645, and XY cutter, and the Duplo PDC, a booklet maker.

Duplo finishers are designed to trim, cut, and fold printed pages according to the commands that are embedded in barcode that is printed on each sheet.

When Duplo support is enabled, YTD 7 automatically translates into barcode the job parameters needed for the finisher. The job can then be printed and fed into the Duplo finisher.

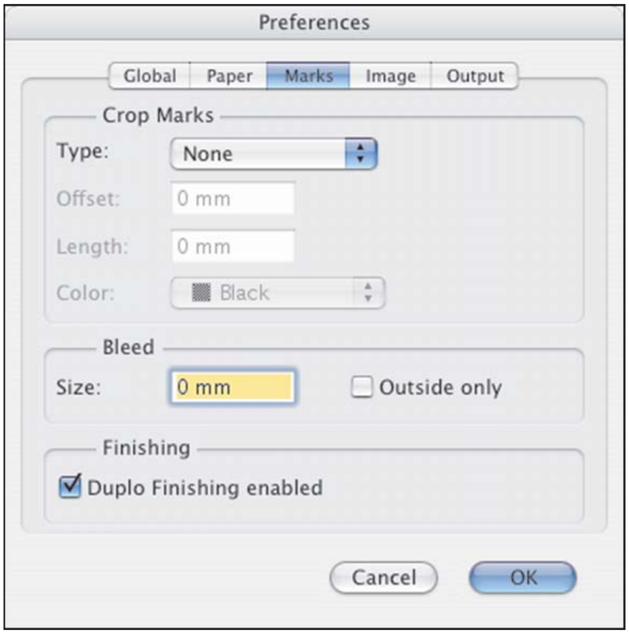
Refer to the *Yours Truly Designer Finishing Options for Duplo finishers How-To guide* if you need to create YTD files that include a barcode for the Duplo finisher. The *How-to guide* is available at the M*y HP Indigo* web site, and contains a description of all the YTD Duplo-related options and step-by-step procedures to help you create your YTD-Duplo job in QuarkXPress.

Enabling Duplo finishing

To enable Duplo finishing options in YTD 7:

1 Select **YTD Preferences** in the YTD menu. The *Preferences* window opens.

2 Click the **Marks** tab.



Crop Marks

Type: None

Offset: 0 mm

Length: 0 mm

Color: ▨ Black

Bleed

Size: 0 mm ☐ Outside only

Finishing

☑ Duplo Finishing enabled

Figure 9-1. ***Preferences* window**

3 In the Finishing pane, check **Duplo Finishing enabled**.

4 Click **OK**. The Duplo finishing feature is enabled and the Finishing tab appears in the *Imposition* window.

5 Refer to the procedures described in the *Yours Truly Designer Finishing Options for Duplo finishers How-To guide*, available from the My HP Indigo web site.

A Service and support

To obtain service, please contact the customer care center (CCC) within your country/region:

Europe

Germany:	+49 (0) 6995307080
France:	+33 (0) 149932498
UK:	+44 (0) 2072950038
Italy:	+39 0 238591081
Belgium:	+32 (0) 26264803
Netherlands	+31 (0) 43 3565900
Luxembourg:	+352 (0) 2730 2067
Ireland:	+353 (0) 1 605 8409
Distribution Channels (DC):	+31 (0) 20 6545543
North America:	1-800-204-6344
Israel:	+972 8 938 1818

North America
Hewlett-Packard Company
Indigo Division
165 Dascomb Road
Andover, MA 01810-5897
USA

International
Hewlett-Packard Company
Indigo Division
Limburglaan 5
6221 SH Maastricht
The Netherlands

Israel
Hewlett-Packard Company
Indigo Division
Kiryat Weizmann
P.O. Box 150
Rehovot 76101, Israel

Index

This is an HP Indigo digital print. Printed in Israel.

www.hp.com/go/indigo

reorder P/N: CA294-00814

invent

Exhibit 1
to Maloney Declaration

WhatTheyThink?

New HP Indigo Press and DFE Enable Cenveo To Expand Digital Print Capabilities

Wednesday, September 19, 2007

PALO ALTO, Calif., September 18, 2007 - HP today announced that Cenveo, the third-largest graphic communications provider in North America and a leader in the management and distribution of print and related products and services, has expanded its end-to-end digital direct-mail and marketing collateral production operation using HP graphic arts technologies.

The Cenveo Direct Mail Group added an HP Indigo press 5000 and an HP Indigo Production Manager digital front end (DFE) to its Nashville, Tenn., facility enabling the firm to dramatically increase its digital print production capabilities while also enhancing its print-on-demand marketing collateral management program.

HP, through its Capture business development program, is working with the Cenveo sales operation to help generate additional sales based on the Direct Mail Group’s expanded digital printing operations.

“HP’s go-to-market strategy in the graphic arts industry is designed to help customers capture more business and expand their opportunities with digital,” said Rich Raimondi, vice president and general manager, U.S. Graphic Arts Business, HP Imaging and Printing Group. “Beyond providing superior image quality and productivity, HP is actively working with customers like Cenveo to help them develop the best digital printing business opportunities.”

Maximizing profit and productivity in marketing collateral and direct mail

Working with financial, health care, and telecommunications companies, Cenveo Nashville offers a comprehensive print-on-demand marketing collateral program saving clients much of the cost, expense and waste associated with offset-printed marketing collateral management. The firm’s variable-data-printing operation drives a high-volume personalized direct-mail effort that gives clients a greater return on investment in terms of cost per response. Overall, the robust processing capabilities of the HP Indigo Production Manager DFE and the reliable, high-volume capabilities of the 5000 model press provide greater throughput, production power and efficiencies for a fast-growing part of Cenveo’s business.

Prior to the installation of the HP Indigo Production Manager, some of Cenveo’s larger and more complex customer files would take up to six hours to RIP. The new HP Indigo Production Manager enables dramatically improved processing times, while the new press allows the company to increase its throughput without adding an additional operator.

“The Cenveo Direct Mail Group has seen our business grow, and we are developing opportunities to offer more to our customers while handling more complex communications needs,” said Steve Kouroupas, general manager, Cenveo Direct Mail Group. “That is really why we wanted to get the latest digital technology from HP.”

Exhibit 2
to Maloney Declaration

Press Release: August 24, 2011
Topics:

Cenveo Elevates Print Publishing with HP

Commercial printing giant meets burgeoning demand with HP Indigo W7200 Digital Press

PALO ALTO -- HP today announced that Cenveo, one of the world's largest providers of print and related resources, has upgraded its short-run publishing and customized collateral capabilities by purchasing an HP Indigo W7200 Digital Press.

Installed recently at one of its core U.S. digital production facilities in Hurlock, Md., the new press enables Cenveo to:

- reduce upfront costs for scientific, technical and medical journal publishers, replacing longer-run offset print and warehousing operations with offset-quality, medium-run-length print-on-demand fulfillment;
- support rapid growth in high-value, fully personalized marketing collateral; and
- sell more customized hybrid production jobs requiring a combination of offset and digital print production, using the W7200 to replicate the color quality of offset lithography.

"Our customers want to save money through just-in-time production and the marketing benefits of variable-data personalization, without sacrificing the level of printing they get with traditional offset," said Cappy Childs, president, Digital, Logistics and Sales divisions, Cenveo. "The HP Indigo W7200, with its offset-quality imaging and high productivity, is a step forward for us."

Integration into streamlined, end-to-end workflows

Cenveo wanted the "latest and greatest" for its Hurlock facility, and the W7200 advances the company's offerings while integrating into its highly productive prepress, digital print and finishing workflow. Prepress color management and press fingerprinting at Cenveo plants are based on the G7® method, and the HP Indigo W7200 digital press's broad color gamut conforms closely to the color quality and repeatability metrics Cenveo maintains in its commercial print production facilities.

Cenveo has extensive experience in digital color printing: its Hurlock facility has run digital color presses from HP and others for the past five years, and the broader Cenveo organization includes digital color presses from every leading manufacturer.

"An organization like Cenveo, with its years of expertise in digital color printing, knows what it wants when it comes to installing an offset-quality digital press that can deliver results," said Jan Riecher, vice president and general manager, Graphics Solutions Business – Americas, HP. "The HP Indigo W7200 brings more value to the publishers and marketers Cenveo serves."

Along with the press, Cenveo purchased an HP SmartStream Production Pro Print Server, a 64-bit digital front-end controller that streamlines data processing and integrates with Cenveo's sophisticated and tightly color-managed prepress workflow. The company also installed an in-line cutting and finishing solution from Hunekeler to further streamline its journal production operations.

HP Indigo imaging: maximizing quality and value

Designed to meet offset-quality, application-focused production needs, the HP Indigo W7200 is the most productive Indigo press model, printing up to 7.5 million pages per month. Capable of printing up to 240 four-color, or up to 960 monochrome, letter-size pages per minute at resolutions up to 2,400 x 2,400 dots per inch, the press is an ideal solution for color publishing applications, transactional/transpromotional printing, direct marketing work and photobook/photo specialty production.

More information about HP Indigo digital presses is available at www.hp.com/go/gsb or by following the HP Graphic Arts Twitter feed, www.twitter.com/hpgraphicarts.

About Cenveo

Cenveo (NYSE: CVO), headquartered in Stamford, Conn., is a leading global provider of print and related resources, offering world-class solutions in the areas of envelopes, custom labels, specialty packaging, commercial print, publisher solutions and business documents. The company provides a one-stop offering through services ranging from design and content management to fulfillment and distribution. With approximately 10,000 employees worldwide, the company prides itself on delivering quality solutions and service every day for customers. For more information please visit Cenveo at www.cenveo.com.

About HP

HP creates new possibilities for technology to have a meaningful impact on people, businesses, governments and society. The world's largest technology company, HP brings together a portfolio that spans printing, personal computing, software, services and IT infrastructure at the convergence of the cloud and connectivity, creating seamless, secure, context-aware experiences for a connected world. More information about HP (NYSE: HPQ) is available at http://www.hp.com.

G7 is a registered trademark of IDEAlliance.

This news release contains forward-looking statements that involve risks, uncertainties and assumptions. If such risks or uncertainties materialize or such assumptions prove incorrect, the results of HP and its consolidated subsidiaries could differ materially from those expressed or implied by such forward-looking statements and assumptions. All statements other than statements of historical fact are statements that could be deemed forward-looking statements, including but not limited to statements of the plans, strategies and objectives of management for future operations; any statements concerning expected development, performance or market share relating to products and services; any statements regarding anticipated operational and financial results; any statements of expectation or belief; and any statements of assumptions underlying any of the foregoing. Risks, uncertainties and assumptions include macroeconomic and geopolitical trends and events; the competitive pressures faced by HP's businesses; the development and transition of new products and services (and the enhancement of existing products and services) to meet customer needs and respond to emerging technological trends; the execution and performance of contracts by HP and its customers, suppliers and partners; the achievement of expected operational and financial results; and other risks that are described in HP's Quarterly Report on Form 10-Q for the fiscal quarter ended April 30, 2011 and HP's other filings with the Securities and Exchange Commission, including but not limited to HP's Annual Report on Form 10-K for the fiscal year ended October 31, 2010. HP assumes no obligation and does not intend to update these forward-looking statements.

Media contacts

Jill Peters, HP
jill.peters@hp.com

David Lindsay, Porter Novelli for HP
david.lindsay@porternovelli.com

About HP Inc.

HP Inc. creates technology that makes life better for everyone, everywhere. Through our portfolio of printers, PCs, mobile devices, solutions, and services, we engineer experiences that amaze. More information about HP Inc. is available at http://www.hp.com.

Exhibit 3
to Maloney Declaration

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

INDUSTRIAL PRINT TECHNOLOGIES LLC,	)	
	)	
Plaintiff,	)	
	)	
v.	)	Case No. 2:14-CV-48- JRG
	)	
O'NEIL DATA SYSTEMS, INC., AND HEWLETT-PACKARD COMPANY,	))	
	)	
Defendants.	)	
	)	

PLAINTIFF INDUSTRIAL PRINT TECHNOLOGIES' DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS

In accordance with Patent Local Rules 3-1 and 3-2, plaintiff Industrial Print Technologies LLC ("IPT") submits its Disclosure of Asserted Claims and Infringement Contentions as to Defendants O'Neil Data Systems, Inc. ("O'Neil") and Hewlett-Packard Company ("HP") (collectively, "Defendants").

1. Right to Supplement

IPT bases these disclosures on its current knowledge, understanding and belief as to the facts and information available to it as of the date of these disclosures. This case is not yet in discovery, and IPT has not yet completed its investigation, collection of information, discovery or analysis related to this action. Accordingly, IPT reserves the right to supplement, amend or modify the information contained herein and to use and introduce such information and any subsequently-identified information at trial. In particular, IPT reserves its right to amend and supplement its identification of asserted claims and modify its identification of accused products and instrumentalities. Additionally, as further discovery is taken, and additional details are provided regarding Defendants' activities, IPT's infringement charts and contentions may need to be amended, supplemented and/or modified. IPT also reserves its right to supplement its disclosure of documents based upon further investigation and discovery.

These disclosures are based at least in part upon IPT's present understanding of the meaning and scope of the claims of the patents-in-suit, in the absence of additional claim construction proceedings or discovery. IPT reserves the right to seek leave to supplement or amend these disclosures if its understanding of the claims changes, including if the Court construes them.

2. Asserted Claims

In accordance with Patent LR 3-1(a), based on information presently available to IPT, IPT states that Defendants infringe:

U.S. Patent No. 5,729,665 ("the '665 patent"), claims 1, 12, 13, and 20;

U.S. Patent No. 5,937,153 ("the '153 patent"), claims 1, 3-5, and 6;

U.S. Patent No. 7,274,479 ("the '479 patent"), claims 9, 10, 15, and 17-19; and

U.S. Patent No. 7,333,233 ("the '233 patent"), claims 12 and 14.

IPT reserves the right to assert additional claims against Defendants based upon results of discovery and further investigation.

3. Accused Instrumentalities and Comparison To Asserted Claims

In accordance with Patent LR 3-1(b), based on information presently available to IPT, Defendant O'Neil has been and is engaged in infringing activities using variable data enabled high-speed printing presses supplied by Defendant HP. Specifically, O'Neil is engaged in infringing the asserted method claims through its use of HP's high-speed printing presses that process variable data print jobs, including HP's Inkjet Web Presses (including for example at least T200, T300, T350 and T400 presses) and its Indigo Digital Presses (including for example at least W3250, 3550, WS4600, 5000, 5600, WS6600, WS6600p, W7250, 7500, 7600, 10000, 20000, and 30000 presses).

To the extent that any steps of the methods covered by the asserted patent claims are performed by third-parties, such as O'Neil's customers and/or their print media agents, Plaintiff alleges that O'Neil is liable for direct infringement because it directs and controls any such third-party steps, including, for example, by dictating the manner by which the third-parties must supply data to enable variable data print jobs to be run on O'Neil's variable data enabled high-speed printing presses, such that O'Neil is jointly and severally and/or vicariously liable for any acts performed by such third-parties on behalf of O'Neil. Upon information and belief, O'Neil provides an Internet website portal through which it provides its products and services to third-party customers and their print media agents. The website portal and/or instructions provided through the website portal directs these third-parties to provide print specification files such that O'Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, O'Neil enters contracts with these third parties, through which O'Neil enforces the obligations that it imposes upon third-parties.

O'Neil has also induced, and continues to induce, these third parties' direct infringement of the asserted claims pursuant to pursuant to 35 U.S.C. § 271(b) by providing the Internet website portal through which it provides its products and services to third-party customers and their print media agents, together with instructions directing third-parties' use of print specification files. Despite its awareness of the asserted claims and of the technology claimed within the asserted claims, O'Neil has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the asserted patent claims and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the asserted patent claims.

HP directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses and Indigo Digital Presses. HP, directly and/or through its subsidiaries, affiliates, agents, and/or business partners has also induced and continues to induce O'Neil's direct infringement of the asserted claims pursuant to 35 U.S.C. § 271(b) by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the asserted claims. HP has also supplied related training and support materials and services. Despite its awareness of the asserted claims and of the technology claimed within the asserted claims, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the asserted patent claims and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the asserted patent claims.

In accordance with Patent LR 3-1(c), IPT provides the following charts, attached as Appendices A and B, which identify specifically where each element and/or step of each asserted claim is found within the Defendants. IPT reserves the right to amend, supplement and modify its contentions and charts based on additional infromation identified through discovery.

4. Literal and Equivalents Infringement

In accordance with Patent LR 3-1(d), as supported and explained in the attached Exhibits, it is currently believed that each of the elements of each of the asserted claims is met literally, and if any claim or claim limitation is not met literally, then it is met under the doctrine of equivalents.

It is expected that the same facts upon which IPT's literal infringement claim is based will also form the basis of IPT's doctrine of equivalents claim, as any differences between the limitations of the asserted claims and the accused products are insubstantial. With respect to the doctrine of equivalents, however, as Defendants have not yet provided details of their non-infringement positions, IPT reserves the right to present further facts to support an assertion of infringement under the doctrine of equivalents.

5. Priority Date

In accordance with Patent LR 3-1(e), IPT alleges that each asserted claim of all four asserted patents is entitled to a priority date at least as early as January 18, 1995, which is the filing date of U.S. Patent No. 5,729,665, to which the patents claim priority.

The subject matter of the asserted claims of the asserted patents was conceived of prior to the filing of the application that became the ‘665 patent.

IPT believes that the subject matter of the asserted claims was conceived of at least as early as 1988, and no later than 1989. The subject matter of the asserted claims was then diligently reduced to practice through the first operating prototype that was completed on or about February 10, 1994. IPT thus contends that the claims are entitled to an invention date during 1989. There was constructive reduction to practice on January 18, 1995. To the extent that further investigation and discovery permits a more specific invention date to be confirmed, IPT will update its disclosures as appropriate.

6. Documents

IPT has made a reasonable investigation for documents identified in P.R. 3-2. Such non-privileged documents are being produced herewith**.**

In accordance with Patent LR 3-2, IPT's documents corresponding to P.R. 3-2(a) include at least those numbered:

TES002976-TES002980, TES004201-TES004202, TES004207-TES004209, TES004210-TES004211, TES004212-TES004245, TES004250-TES004278, TES004279-TES004280, TES004281-TES004282, TES004283-TES004284, TES004320-TES004324, TES004325-TES004330, TES004331-TES004333, TES004415-TES004415, TES004416-TES004416, TES004812-TES004812, TES004813-TES004814, TES004822-TES004827, TES004828-TES004833, TES004834-TES004838, TES004843-TES004844, TES004847-TES004848, TES004858-TES004860, TES004861-TES004863, TES004864-TES004866, TES004867-TES004869, TES005505-TES005521, TES005522-TES005527, TES009900-TES010246, TES011201-TES011202, TES013273-TES013304, TES013477-TES013478, TES015782-TES015786, TES018684-TES018720, TES036025-TES036138, TES107224-TES107234, TES108742-TES108775, TES108776-TES108798, TES108799-TES108821, TES237440-TES237442, TES240475-TES240608.

IPT's documents corresponding to P.R. 3-2(b) include at least those numbered:

TES002250-TES002253, TES002269-TES002271, TES002305-TES002412, TES002870-TES002873, TES003038-TES003047, TES003856-TES003857, TES003858-TES003860, TES003861-TES003864, TES003865-TES003865, TES003867-TES003878, TES003879-TES003902, TES003903-TES003918, TES003919-TES003925, TES003926-TES003964, TES003965-TES003981, TES003982-TES003985, TES003986-TES003993, TES003998-TES003999, TES004000-TES004000, TES004001-TES004001, TES004077-TES004078, TES004083-TES004084, TES004085-TES004086, TES004087-TES004087, TES004088-TES004091, TES004092-TES004092, TES004093-TES004093, TES004094-TES004094, TES004095-TES004095, TES004096-TES004096, TES004097-TES004097, TES004099-TES004100, TES004104-TES004104, TES004119-TES004119, TES004120-TES004120, TES004123-TES004126, TES004127-TES004130, TES004131-TES004131, TES004132-TES004132, TES004133-TES004133, TES004134-TES004134, TES004135-TES004135, TES004136-TES004138, TES004139-TES004139, TES004140-TES004140, TES004141-TES004141, TES004142-TES004144, TES004145-TES004146, TES004148-TES004148, TES004149-TES004151, TES004152-TES004154, TES004155-TES004156, TES004157-TES004163, TES004184-TES004185, TES004186-TES004197, TES004207-TES004209, TES004210-TES004211, TES004212-TES004245, TES004250-TES004278, TES004279-TES004280, TES004281-TES004282, TES004283-TES004284, TES004286-TES004286, TES004287-TES004287, TES004288-TES004288, TES004289-TES004289, TES004290-TES004290, TES004291-TES004291, TES004293-TES004293, TES004294-TES004295, TES004296-TES004302, TES004303-TES004303, TES004305-TES004305, TES004306-TES004306, TES004307-TES004307, TES004308-TES004308, TES004309-

TES004309, TES004310-TES004310, TES004320-TES004324, TES004325-TES004330, TES004331-TES004333, TES004365-TES004365, TES004366-TES004366, TES004367-TES004367, TES004368-TES004396, TES004397-TES004397, TES004398-TES004398, TES004403-TES004403, TES004404-TES004404, TES004405-TES004405, TES004409-TES004409, TES004417-TES004417, TES004418-TES004418, TES004419-TES004419, TES004420-TES004420, TES004445-TES004445, TES004446-TES004446, TES004447-TES004451, TES004452-TES004452, TES004453-TES004453, TES004454-TES004454, TES004455-TES004455, TES004478-TES004478, TES004480-TES004480, TES004481-TES004481, TES004482-TES004482, TES004483-TES004483, TES004486-TES004486, TES004487-TES004487, TES004489-TES004489, TES004490-TES004490, TES004491-TES004491, TES004492-TES004492, TES004493-TES004493, TES004494-TES004494, TES004495-TES004496, TES004497-TES004497, TES004498-TES004499, TES004500-TES004500, TES004501-TES004501, TES004502-TES004502, TES004503-TES004504, TES004505-TES004505, TES004506-TES004506, TES004507-TES004508, TES004509-TES004509, TES004510-TES004510, TES004511-TES004511, TES004512-TES004512, TES004513-TES004513, TES004514-TES004515, TES004516-TES004516, TES004517-TES004518, TES004519-TES004520, TES004521-TES004521, TES004522-TES004523, TES004525-TES004525, TES004526-TES004528, TES004529-TES004530, TES004531-TES004537, TES004538-TES004539, TES004540-TES004545, TES004551-TES004551, TES004579-TES004580, TES004581-TES004581, TES004582-TES004582, TES004583-TES004584, TES004585-TES004585, TES004586-TES004591, TES004592-TES004592, TES004593-TES004598, TES004599-TES004602, TES004603-TES004605, TES004606-TES004607, TES004614-TES004615, TES004616-TES004620, TES004621-TES004621, TES004622-TES004625, TES004626-TES004627, TES004628-TES004631, TES004632-TES004636, TES004637-TES004648, TES004649-TES004652, TES004653-TES004656, TES004657-TES004664, TES004665-TES004665, TES004669-TES004673, TES004674-TES004674, TES004675-TES004677, TES004678-TES004691, TES004692-TES004698, TES004699-TES004700, TES004701-TES004710, TES004711-TES004714, TES004715-TES004717, TES004724-TES004725, TES004726-TES004729, TES004731-TES004734, TES004735-TES004735, TES004736-TES004738, TES004739-TES004739, TES004740-TES004756, TES004757-TES004758, TES004759-TES004773, TES004774-TES004774, TES004775-TES004794, TES004795-TES004795, TES004796-TES004796, TES004797-TES004797, TES004798-TES004798, TES004812-TES004812, TES004813-TES004814, TES004816-TES004821, TES004822-TES004827, TES004828-TES004833, TES004834-TES004838, TES004840-TES004840, TES004841-TES004841, TES004842-TES004842, TES004843-TES004844, TES004845-TES004846, TES004847-TES004848, TES004849-TES004849, TES004850-TES004850, TES004851-TES004851, TES004852-TES004852, TES004853-TES004853, TES004854-TES004855, TES004856-TES004857, TES004858-TES004860, TES004861-TES004863, TES004864-TES004866, TES004867-TES004869, TES004870-TES004871, TES004880-TES004890, TES004891-TES004896, TES004897-TES004938, TES004939-TES004939, TES004940-TES004946, TES004947-TES004958, TES004959-TES005014, TES005015-TES005017, TES005018-TES005018, TES005019-TES005019, TES005020-TES005020, TES005028-TES005028, TES005029-TES005030, TES005031-TES005032, TES005033-TES005033, TES005034-TES005035, TES005036-TES005039, TES005040-TES005053, TES005054-TES005061, TES005062-TES005062, TES005063-TES005065, TES005066-TES005066, TES005067-TES005071, TES005072-

TES005073, TES005074-TES005074, TES005075-TES005075, TES005076-TES005079, TES005080-TES005089, TES005090-TES005095, TES005096-TES005096, TES005097-TES005098, TES005099-TES005099, TES005107-TES005110, TES005111-TES005114, TES005115-TES005121, TES005122-TES005127, TES005128-TES005135, TES005136-TES005145, TES005146-TES005150, TES005151-TES005159, TES005160-TES005168, TES005169-TES005174, TES005175-TES005183, TES005184-TES005210, TES005211-TES005216, TES005217-TES005225, TES005226-TES005252, TES005254-TES005254, TES005255-TES005255, TES005256-TES005256, TES005257-TES005257, TES005258-TES005258, TES005259-TES005259, TES005260-TES005260, TES005261-TES005265, TES005266-TES005266, TES005267-TES005267, TES005268-TES005270, TES005271-TES005272, TES005273-TES005278, TES005279-TES005290, TES005457-TES005457, TES005458-TES005458, TES005460-TES005463, TES005464-TES005466, TES005467-TES005468, TES005469-TES005470, TES005505-TES005521, TES005528-TES005528, TES005532-TES005532, TES005533-TES005533, TES005534-TES005536, TES005537-TES005539, TES005540-TES005540, TES005564-TES005565, TES005566-TES005569, TES005570-TES005573, TES005574-TES005579, TES005580-TES005583, TES005584-TES005585, TES005586-TES005586, TES005587-TES005587, TES005588-TES005588, TES005589-TES005589, TES005590-TES005590, TES005591-TES005591, TES005598-TES005598, TES005599-TES005599, TES005600-TES005600, TES005601-TES005604, TES005605-TES005607, TES005609-TES005609, TES005610-TES005610, TES005611-TES005639, TES005640-TES005640, TES005641-TES005645, TES005672-TES005674, TES005675-TES005677, TES005678-TES005680, TES006504-TES006653, TES006695-TES006695, TES006723-TES006724, TES006816-TES006846, TES007208-TES007223, TES008359-TES008364, TES008365-TES008377, TES008378-TES008383, TES008384-TES008395, TES008398-TES008408, TES008409-TES008417, TES008418-TES008420, TES008421-TES008430, TES008431-TES008432, TES008433-TES008435, TES008436-TES008438, TES008439-TES008443, TES008444-TES008448, TES008463-TES008464, TES008465-TES008466, TES008467-TES008473, TES008474-TES008503, TES008504-TES008505, TES008506-TES008524, TES008525-TES008528, TES008529-TES008543, TES008544-TES008551, TES008552-TES008554, TES008555-TES008584, TES008614-TES008617, TES008618-TES008620, TES008650-TES008654, TES008655-TES008680, TES008691-TES008695, TES009442-TES009503, TES009525-TES009537, TES009595-TES009595, TES009659-TES009662, TES009848-TES009899, TES011141-TES011142, TES011143-TES011147, TES011148-TES011153, TES011154-TES011157, TES011158-TES011168, TES011169-TES011179, TES011180-TES011185, TES011186-TES011198, TES011199-TES011200, TES011203-TES011204, TES011205-TES011236, TES011237-TES011241, TES011242-TES011247, TES011248-TES011249, TES011250-TES011256, TES011257-TES011258, TES011259-TES011263, TES011264-TES011265, TES011266-TES011267, TES011268-TES011274, TES011275-TES011276, TES011277-TES011278, TES011279-TES011280, TES011281-TES011282, TES011283-TES011284, TES011285-TES011288, TES011289-TES011294, TES011295-TES011303, TES011310-TES011372, TES011608-TES011656, TES011657-TES011669, TES011817-TES011820, TES011823-TES011824, TES011836-TES011839, TES011840-TES011850, TES011851-TES011861, TES011862-TES011867, TES011868-TES011881, TES011882-TES011883, TES011884-TES011885, TES011886-TES011887, TES011888-TES011919, TES011920-TES011924, TES011925-TES011930, TES011931-TES011932, TES011933-TES011939, TES011940-

TES011941, TES011942-TES011946, TES011947-TES011948, TES011949-TES011950, TES011951-TES011957, TES011958-TES011959, TES011960-TES011961, TES011962-TES011963, TES011964-TES011965, TES011966-TES011967, TES011968-TES011971, TES011972-TES011977, TES011978-TES011986, TES012004-TES012010, TES012011-TES012014, TES012040-TES012041, TES012042-TES012047, TES012048-TES012054, TES012290-TES012292, TES012293-TES012341, TES012342-TES012354, TES013081-TES013142, TES013143-TES013161, TES013162-TES013174, TES013273-TES013304, TES014021-TES014090, TES014091-TES014151, TES014190-TES014227, TES014228-TES014329, TES014330-TES014391, TES014392-TES014438, TES014439-TES014472, TES014473-TES014479, TES014480-TES014537, TES014538-TES014605, TES014606-TES014662, TES014663-TES014766, TES014767-TES014841, TES014842-TES014907, TES014908-TES014979, TES014980-TES015120, TES015121-TES015304, TES015787-TES015793, TES015794-TES015796, TES015797-TES015799, TES015810-TES015813, TES016292-TES016334, TES018613-TES018623, TES018626-TES018679, TES019295-TES019317, TES019318-TES019340, TES019341-TES019345, TES019346-TES019351, TES019356-TES019357, TES019358-TES019379, TES022843-TES022853, TES023472-TES023476, TES025611-TES025621, TES025622-TES025624, TES025626-TES025636, TES025637-TES025643, TES025644-TES025670, TES025671-TES025672, TES025673-TES025679, TES032626-TES032627, TES032628-TES032629, TES032630-TES032657, TES032664-TES032667, TES032668-TES032676, TES032677-TES032688, TES032689-TES032695, TES038176-TES038282, TES038419-TES038585, TES038623-TES038694, TES038829-TES038951, TES038952-TES039181, TES040237-TES040526, TES040784-TES040969, TES040970-TES041088, TES041343-TES041422, TES047510-TES047514, TES100247-TES100251, TES100286-TES100287, TES100293-TES100326, TES100580-TES100580, TES100604-TES100604, TES100605-TES100608, TES100609-TES100609, TES107224-TES107234, TES274326-TES274326, TES279177-TES279177, TES280365-TES280365, TES280374-TES280374, TES281386-TES281386, TES281730-TES281730, TES281739-TES281739, TES281747-TES281747,.

IPT’s documents corresponding to P.R. 3-2(c) are numbered:

TES336688-TES336813, TES337205-TES337279, TES337507-TES337622, TES337623-TES337713, TES338116-TES338285, TES338286-TES338324, TES340745-TES342864, TES342865-TES344969, TES344970-TES347044, TES347045-TES349151, TES349455-TES352270, TES352271-TES355288.

Date: April 7, 2014

/s/ Alison Aubry Richards
Timothy P. Maloney (IL 6216483)
Alison Aubry Richards (IL 6285669)
Nicole L. Little (IL 6297047)
David A. Gosse (IL 6299892)
Fitch, Even, Tabin & Flannery
120 South LaSalle Street, Suite 1600
Chicago, Illinois 60603
Telephone: (312) 577-7000
Facsimile: (312) 577-7007

T. John Ward, Jr.
Texas State Bar No. 00794818
Email: jw@wsfirm.com
Wesley Hill
Texas State Bar No. 24032294
Email: wh@wsfirm.com
WARD & SMITH
Post Office Box 1231
Longview, TX 75606
Telephone: (903) 757-6400
Facsimile: (903) 757-2323

Steven C. Schroer (IL 6250991)
scschr@fitcheven.com
FITCH, EVEN, TABIN & FLANNERY
1942 Broadway
Suite 213
Boulder, CO 80302
Telephone: 303.402.6966
Facsimile: 303.402.6970

Counsel for Plaintiff

CERTIFICATE OF SERVICE

The undersigned certifies that a copy of the above document PLAINTIFF IPT'S DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS and exhibits was sent by email and first class mail to the counsel of record below on this April 7, 2014:

Evan Budaj
Edward R. Reines
Weil, Gotshal & Manges LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065-1134
evan.budaj@weil.com
edward.reines@weil.com

/s/ Alison Aubry Richards
Alison Aubry Richards
Attorney for Plaintiff

Exhibit A

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. Discovery in this case has not yet commenced, and the charts below do not reflect any information produced by defendants O'Neil Data Systems, Inc. or Hewlett-Packard Company. IPT reserves the right to support its theories with additional material produced by the defendants or subsequently identified by IPT.

[1] O'Neil Data Solutions website http://www.oneildata.com/services/onesuite/onedms

[2] HP, O'Neil Data Systems: HP Indigo Presses Power Targeted Marketing Campaigns, available at http://h10088.www1.hp.com/gap/download/O_Neil_Data_Systems_HP_Indigo_presses_Case_Study.pdf

[3] O'Neil Data Systems and the HP T400 Spearhead Industry Change, available at http://www.oneildata.com/hp-large-format-printing/oneil-data-systems-and-the-hp-t400-spearhead-industry-change/

[4] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf

[4a] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf

[5] HP T200 Data Sheet http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=4AA3-0798ENW

[6] HP T300 Data Sheet http://h10088.www1.hp.com/gap/download/products/T300-Color-Inkjet-Web-Press/WebPress_IHPS_DS_US.PDF

[7] HP T350 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T350_US.pdf

[8] HP T400 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T400_US.pdf

[9] HP Indigo w3250 Data Sheet http://ccserver.copiercatalog.com/catalogfiles/HP_Indigo_w3250_sales1.pdf

[10] HP Indigo 5000 Data Sheet http://h10088.www1.hp.com/gap/Data/en/us/5000_DS_Low.pdf

[11] HP Indigo 7500 Data Sheet http://www.csi2.com/resources/HP_Indigo_7500.pdf

[12] PPML Template available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html

[13] PPML Specification v1.5 PDF available at http://www.standards.podi.org/ppml/specification.html

[14] PPML Specification v2.1 PDF available at http://www.standards.podi.org/ppml/specification.html

[15] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf

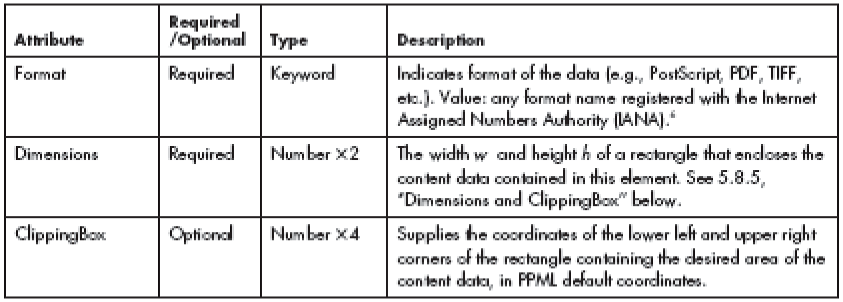
[16] HP Indigo Yours Truly Designer 7 User Guide (attached)

[17] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.

U.S. Patent No. 5,729,665 ("the '665 patent")

'665 Patent, Claim 1	
1. A method for generating multiple bit maps suitable for high-speed printing or plate-making comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below.
(a) generating a page description code representing a template, said page description code defining at least one variable data area and said page description code further defining a graphics state corresponding to said variable data area, said graphics state	O'Neil's OneSuite^{TM} website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. To the extent that third-parties, such as O'Neil's customers and/or their print media agents, perform the step of generating these files, O'Neil directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. The OneSuite^{TM} website portal and/or instructions provided through the website portal directs

'665 Patent, Claim 1	
including at least one attribute which controls the appearance of variable data in said variable data area;	third-parties to provide print specification files such that O'Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, upon information and belief, O'Neil enters contracts with these third parties through which O'Neil enforces the obligations that it imposes upon third-parties. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39.

<table>
<tr><th>'665 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.

<table><tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr><tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr><tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr><tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr></table>
Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(b) executing said page description code to generate a bit map of said template, and during said execution, identifying said variable data</td><td>O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server,</td></tr>
</table>

'665 Patent, Claim 1	
area defined by said page description code and reserving said graphics state corresponding to said variable data area upon said identification;	which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. The VDP file also defines a variable data area by including information such as the size and location for each variable data element and includes graphics state information including

'665 Patent, Claim 1	
	appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data.
(c) retrieving variable data;	The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(d) associating said variable data with said graphics state corresponding to said variable data area;	The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field.
(e) applying said graphics state corresponding to said variable data area to said variable data to generate a variable data bit map; and	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content

‘665 Patent, Claim 1

data.

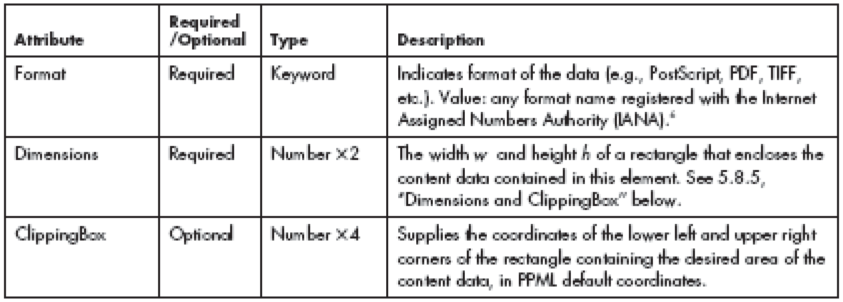
The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [13] at 22; Ref. [14] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The

'665 Patent, Claim 1	
	appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.
(f) merging said variable data bit map with said bit map of said template;	The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at 2. Software running on the printer controller interprets PPML, PPMLT, and JLYT files according to the structures defined for each of these VDP files types. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.
wherein said graphics state corresponding to said variable data area is applied repeatedly to variable data to generate a multitude of variable data bit maps without the need to repeat said executing step (b).	The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document.

'665 Patent, Claim 12	
12. The method of claim 1 wherein said reserving, retrieving, associated, applying, and merging steps are repeated for each variable data area defined by said page description code.	VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

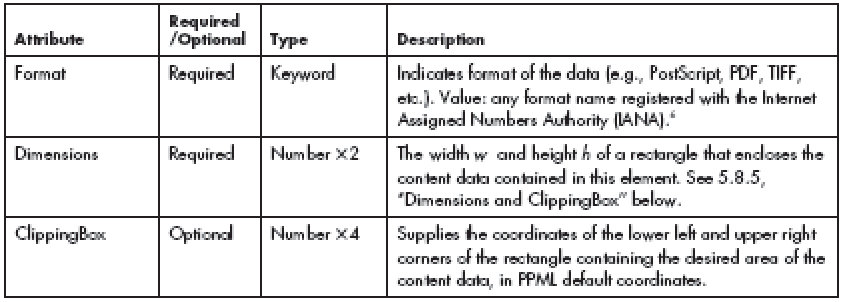
'665 Patent, Claim 13	
13. The method of claim 12 wherein said reserving, retrieving, associating, applying and merging steps are activated by a control task running in a printer controller, and wherein said control task interrupts said page description code execution upon identifying a predetermined command in said	As mentioned earlier, the steps of reserving, retrieving, associating, applying and merging are all activated and monitored by a control task running in the HP printer controller. On information and belief, the control task interrupts said page description code execution upon identifying a predetermined command in said page description code to enable other operations to be performed.

'665 Patent, Claim 13	
page description code.	

'665 Patent, Claim 20	
20. A method for generating a plurality of bit maps suitable for high-speed printing or plate-making from a page description code representing a template and defining at least one variable data area, and from a merge file containing a plurality of data records of at least one variable data field type, the method comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below. O'Neil's OneSuite$^{\text{TM}}$ website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines size and location for static and variable data areas, and further provides appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the

‘665 Patent, Claim 20	
	object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [13] at 22; Ref. [14] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39. “The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.

‘665 Patent, Claim 20



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

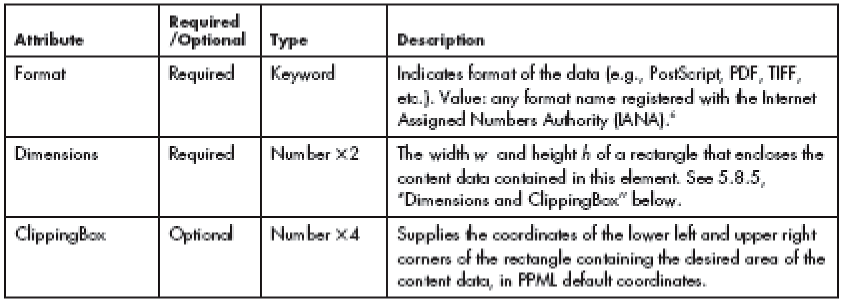
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.

The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24.

Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., “<xsl: value-of select='name'/>” points to a database entry for the “name” element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.

'665 Patent, Claim 20	
executing a page description code interpretive program, said interpretive program generates graphics states for each data area defined by said page description code;	O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including JLYT/SNAP and PPMLT. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which processes PPML, PPMLT, and JLYT files. Ref. [4a]. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and

<table>
<tr><th>‘665 Patent, Claim 20</th><th></th></tr>
<tr><td></td><td>orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.
“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>executing a control task in conjunction with said interpretive program, said</td><td>O’Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O’Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production</td></tr>
</table>

'665 Patent, Claim 20	
control task identifies said variable data area defined by said page description code and reserves said graphics states generated by said interpretive program for said variable data area, said control task generates a template bit map defined by said page description code, and after the completion of said interpretive program, said control task saves said template bit map in memory; and	Manager supports multiple VDP file types including JLYT/SNAP and PPMLT. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such a control task running on these printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files to identify variable data elements by scanning the variable data files and finding the tags associated with such variable data. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. The VDP file also defines a variable data area by including information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. The printer controller is also used to create a template bitmap and store a template bitmap. The

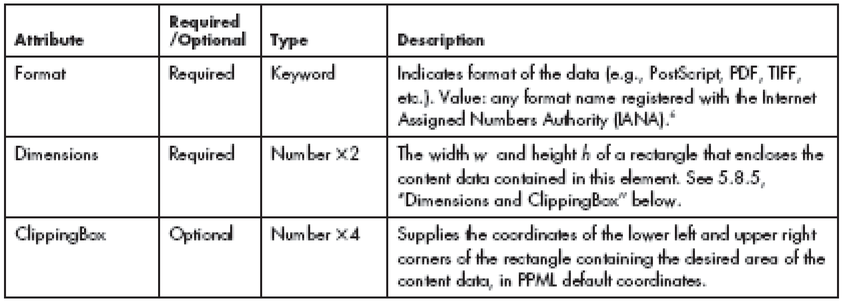
'665 Patent, Claim 20	
	template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
executing a merge task upon completion of said interpretive program, said merge task generates variable data bit maps for said data records in said merge file by applying said reserved graphics states to said data records, and said merge task merges said variable data bit maps with a separate copy of said template bit map to create the plurality of bit maps suitable for high-speed printing or plate making;	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36 "The TRANSFORM element represents a two-dimensional homogeneous transformation

‘665 Patent, Claim 20

matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

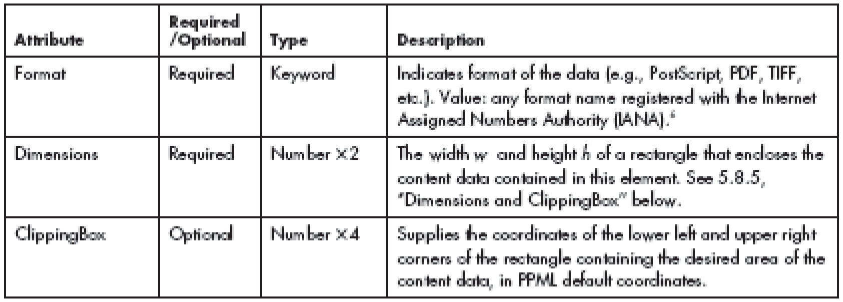
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.

'665 Patent, Claim 20	
	The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at 2. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.
whereby said reserved graphics states are applied repeatedly to said data records to generate said variable data bit maps for said data records without the need to repeat said steps of executing a page description code interpretive program and executing a control task in conjunction with said interpretive program.	The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

U.S. Patent No. 5,937,153 ("the '153 patent")

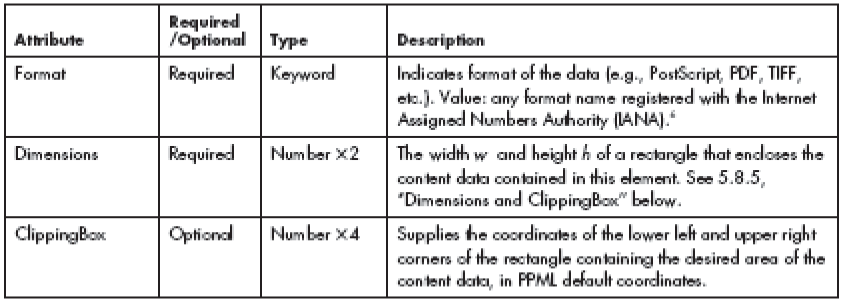
'153 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Ref. [2-11]. Each of these digital presses receives print job information from at least one press controller, as further described below.
(a) generating a page description code specification, the page description code specification defining at least one data area to become variable, and the page description code further defining a graphics state corresponding to the data area, the graphics state including at least one attribute which controls the appearance of data in the data area;	O'Neil's OneSuite^{TM} website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. To the extent that third-parties, such as O'Neil's customers and/or their print media agents, perform the step of generating these files, O'Neil directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. The OneSuite^{TM} website portal and/or instructions provided through the website portal directs third-parties to provide print specification files such that O'Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, upon information and belief, O'Neil enters contracts with these third parties through which O'Neil enforces the obligations that it imposes upon third-parties. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, a PPML file includes a hierarchy

'153 Patent, Claim 1	
	of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.

<table>
<tr><th>'153 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.
</td></tr>
<tr><td>(b) interpreting the page description code specification, and during the interpretation, identifying the data area defined by the page description code specification;</td><td>O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a].
The VDP file defines variable data areas based on the surrounding tags of the data element. Tags within the VDP file include information such as the size and location for each variable data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the</td></tr>
</table>

'153 Patent, Claim 1	
	clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
(c) storing the graphics state corresponding to the data area upon the identification of the variable data area in step (b);	The VDP file includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data.
(d) retrieving a variable data item from a plurality of variable data items;	The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(e) applying the stored graphics state to the variable data item to generate a variable data bit map; and	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the

‘153 Patent, Claim 1	
	appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [13] at 22; Ref. [14] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39. “The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.

<table>
<tr><th>'153 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(f) repeating steps (d) and (e) for remaining variable data items in the plurality of variable data items, whereby the stored graphics state is applied repeatedly to generate a plurality of variable data bit maps.</td><td>The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before</td></tr>
</table>

'153 Patent, Claim 1	
	they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

'153 Patent, Claim 3	
3. The computer implemented method of claim 1, wherein the page description code specification represents a template and includes a static data area; and the computer implemented method further comprises the steps of:	As described for claim 1 of the '153 patent, O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. These VDP files use static data areas to quickly manage VDP jobs. PPML for example, performs more efficiently when the static data areas are defined in advance. Ref. [12] at 10.
executing portions of the page description code specification corresponding to the static data area to generate a template bit map;	O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using

'153 Patent, Claim 3	
	OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
storing the template bit map; and	As described above, the static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
merging each of the plurality of the variable data bit maps into a clean copy of the template bit map to create a plurality of merged bit maps.	The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at 2. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.

'153 Patent, Claim 4	
4. The computer implemented method of claim 1, wherein the identifying step includes the step of detecting predefined	As described for claim 1 of the '153 patent, the controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale

'153 Patent, Claim 4	
characters within a text string defined in the page description code specification.	and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.

'153 Patent, Claim 5	
5. The computer implemented method of claim 1, wherein the attribute is a size attribute, a font attribute, a position attribute, an orientation attribute or a location attribute.	As described above, PPML, PPMLT, and JLYT can each define appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34.

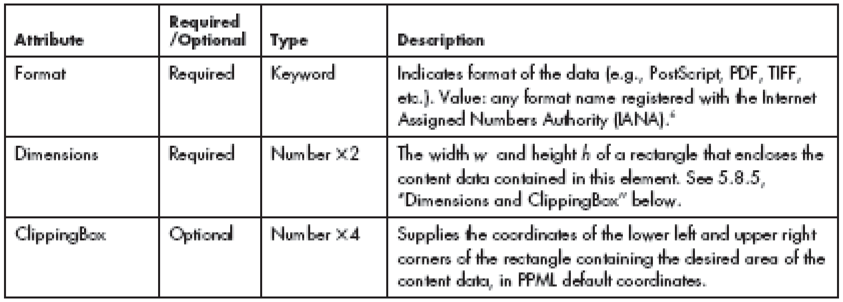
‘153 Patent, Claim 5	

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling,

'153 Patent, Claim 5	
	and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.

'153 Patent, Claim 6	
6. A computer implemented method for processing a page description code specification comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Ref. [2-11]. Each of these digital presses receives print job information from at least one press controller, as further described below. O'Neil's OneSuite^{TM} website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11].
interpreting the page description code specification, and during the interpretation, identifying a data area defined by the page description code specification;	O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer

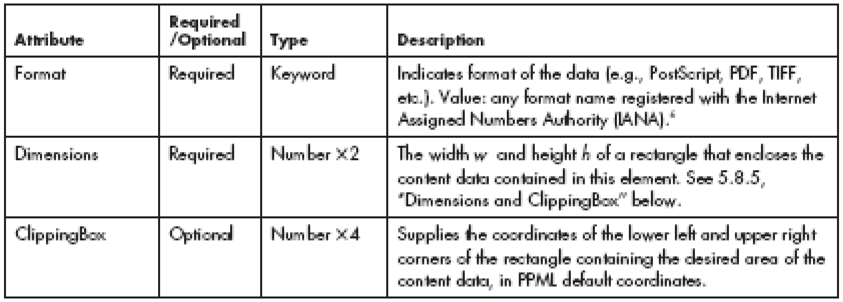
'153 Patent, Claim 6	
	controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
upon the identification of the data area, storing a graphics state set forth in the page description code specification which defines an attribute of how data is to appear in the data area; and	The VDP file defines a variable data area by including information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data.
repeatedly retrieving data records from a plurality of data records and applying the stored graphics state to the data records to generate a plurality of bitmaps of the data records so that the bitmaps of the data	The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT

'153 Patent, Claim 6	
records include the attribute.	scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4. The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field. The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13]

‘153 Patent, Claim 6	

at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number X2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number X4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.

The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document.

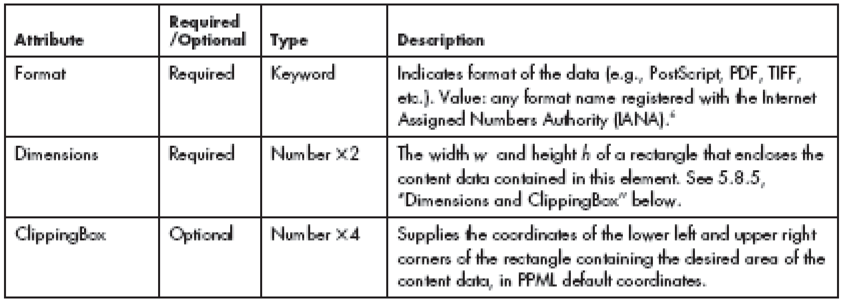
'153 Patent, Claim 6	
	VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

U.S. Patent No. 7,274,479 ("the '479 patent")

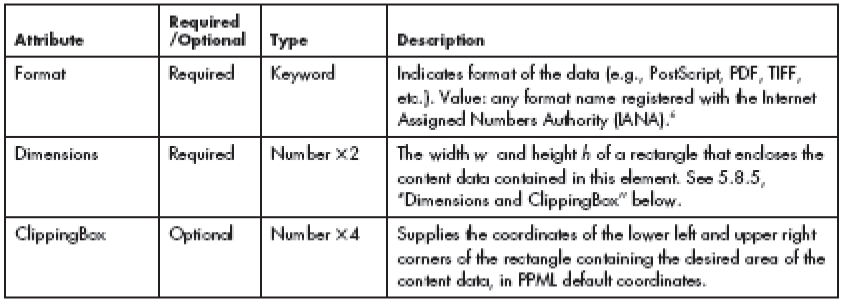
'479 Patent, Claim 9	
9. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing, comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below.
(a) providing a print specification, the print specification defining at least one variable data area and at least one static data area;	O'Neil's OneSuite$^{\text{TM}}$ website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents at least one variable data are and at least one static data area. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. Further, the OBJECT tag within a PPML file

'479 Patent, Claim 9	
	"associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files also include channels that define links to variable content. Ref. [17] at 5.
(b) providing a plurality of variable data items;	The VDP file provides variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(c) identifying the variable data area;	O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files. The controller identifies variable data elements by scanning the variable

'479 Patent, Claim 9	
	data files and finding the tags associated with such variable data, as described above.
(d) associating a graphic state with the variable data area, the graphic state including at least one attribute controlling the appearance of items to be printed in the variable data area;	The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field. Each of the PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36 "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37 "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(e) retrieving a variable data item from the plurality of variable data items;</td><td>The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields</td></tr>
</table>

'479 Patent, Claim 9	
	internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(f) generating a bitmap for the variable item, the generating step including a step of applying the graphic state associated with the variable data area to the variable data item; and	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 54; Ref. [15] at 2. The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36 "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39.
"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.
<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(g) repeating steps (e) and (f) for remaining variable data</td><td>The press controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The press controller creates multiple</td></tr>
</table>

'479 Patent, Claim 9	
items in the plurality of variable data items, whereby the graphic state associated with the variable data area is applied repeatedly to generate a plurality of variable data bitmaps.	variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the press controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

'479 Patent, Claim 10	
10. The method of claim 9, wherein the graphic state associated with the variable data area is defined within the print specification.	Each of the PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data, as discussed above with respect to element (d) of claim 9 of the '479 Patent. The appearance information may be defined within the print specification either by referencing an external file or by providing the appearance information directly within the VDP file.

'479 Patent, Claim 15	
15. The method of claim 9, wherein the variable data area and the static data area are	As described for claim 9 of the '479 Patent, the VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file

'479 Patent, Claim 15	
defined, at least in part, by page description language commands.	"associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.

'479 Patent, Claim 17	
17. The method of claim 9, further comprising a step of caching a representation of the static data area, whereby the cached representation of the static data area is available for merging with the variable data bitmaps to generate merged documents.	The static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2.

'479 Patent, Claim 17	
	IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the printer controller. *See* Ref. [17] at 5.

'479 Patent, Claim 18	
18. The method of claim 17, wherein the cached representation of the static data area is a bitmap representation.	The cached representation of the static data area is a bitmap to avoid the redundant burden of the system to continually compute the contents appearance, as discussed above for claim 17 of the '479 Patent. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7.

'479 Patent, Claim 19	
19. The method of claim 9, wherein: the plurality of data items are associated with a field name; and	As described for claim 9 of the '479 Patent, each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file.
the step of identifying a variable data area includes the step of	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that

'479 Patent, Claim 19	
detecting, in the print specification, a character string associated with the variable data area that matches the field name associated with the plurality of data items.	the controller is processing. For example, the OBJECT tag within a PPML file, when contained within a MARK tag denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.

U.S. Patent No. 7,333,233 ("the '233 patent")

'233 Patent, Claim 12	
12. A computer implemented method for generating a static bitmap suitable for high-speed variable printing, comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below.
providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	O'Neil's OneSuite$^{\text{TM}}$ website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines size and location for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable

'233 Patent, Claim 12	
	Object (section 5.12)." Ref. [13] at 22; Ref. [14] at 34. The OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. Thus, PPMLT files provide at least the same size and location information provided by PPML files, because the PPMLT standard fully incorporates the PPML standard. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In yet another example, JLYT files provide a variety of information to define static and variable data areas. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
interpreting the page description language file, and during the interpreting step, generating a static bitmap of the static data area;	O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is

'233 Patent, Claim 12	
	expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
and saving the static bitmap, whereby the saved static bitmap is used repeatedly in the generation of a plurality of documents, each of which contains the static bitmap and a variable data bitmap.	The static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2.

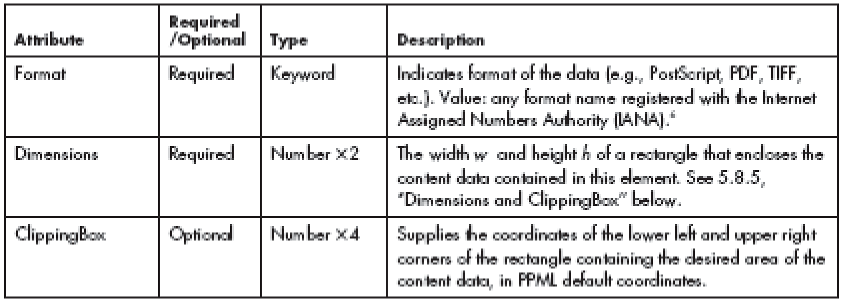
'233 Patent, Claim 14	
14. A computer implemented method for generating a plurality of bitmaps suitable for high-speed printing, comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website

'233 Patent, Claim 14	
	portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below.
(a) providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	O'Neil's OneSuite$^{\text{TM}}$ website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines size and location for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12)." Ref. [13] at 22; Ref. [14] at 34. The OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a

'233 Patent, Claim 14	
	document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. Thus, PPMLT files provide at least the same size and location information provided by PPML files, because the PPMLT standard fully incorporates the PPML standard. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In yet another example, JLYT files provide a variety of information to define static and variable data areas. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
(b) providing a merge file including a plurality of variable data items;	The VDP files can use variable data elements stored internally or in separate files. in PPML documents, variable data is contained within a non-reusable OBJECT tag, which stores data either internally or externally. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(c) processing the page description language file, and during the processing step, generating a static bitmap of the static data area and associating the variable data area with the plurality of variable data items;	O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML,

'233 Patent, Claim 14	
and	PPMLT, and JLYT files; and creates a static bitmap. The static bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. The printer controller and/or the press associates the variable data found in the VDP file to the corresponding variable data area that it retrieved from the file. Each variable data field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field.

'233 Patent, Claim 14	
	The VDP file provides variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(d) saving the static bitmap;	The static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2.
(e) generating a first variable data bitmap of a first one of the variable data items utilizing a graphics state associated with the variable data area;	The printer controller and/or the press applies appearance information to the corresponding variable data to generate a variable data bit map. Ref. [12] at 54; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript,

'233 Patent, Claim 14	
	PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36 "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37 "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40. 

Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

'233 Patent, Claim 14	
	Ref. [13] at 28; Ref. [14] at 40. In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.
(f) merging the first variable data bitmap with a copy of the static bitmap to produce a first output bitmap;	The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at 2. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.
(g) generating a next variable data bitmap of a next one of the variable data items utilizing a graphics state associated with the variable data area;	The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The press controller creates multiple variable data bitmaps, according to the contentions with respect to element (e). Appearance information is reused for each instance of the document. To render each additional variable data record, the printer controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each

'233 Patent, Claim 14	
	instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.
and (h) merging the next variable data bitmap with a copy of the static bitmap to produce a next output bitmap;	The press controller merges the variable data bitmaps with the template bitmap according to the contentions with respect to element (f). The appearance information and the template bitmap are reused for each instance of the document. The template bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. The template bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2.
and (i) repeating steps (g) (h)	The activities performed for steps (g) and (h) are repeated for each remaining variable data item in

'233 Patent, Claim 14	
for remaining variable data items in the plurality of variable data items.	the plurality of data items.

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. Discovery in this case has not yet commenced, and the charts below do not reflect any information produced by defendants O'Neil Data Systems, Inc. or Hewlett-Packard Company. IPT reserves the right to support its theories with additional material produced by the defendants or subsequently identified by IPT.

[1] O'Neil Data Solutions website http://www.oneildata.com/services/onesuite/onedms
[2] HP, O'Neil Data Systems: HP Indigo Presses Power Targeted Marketing Campaigns, available at http://h10088.www1.hp.com/gap/download/O_Neil_Data_Systems_HP_Indigo_presses_Case_Study.pdf
[3] O'Neil Data Systems and the HP T400 Spearhead Industry Change, available at http://www.oneildata.com/hp-large-format-printing/oneil-data-systems-and-the-hp-t400-spearhead-industry-change/
[4] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf
[4a] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf
[5] HP T200 Data Sheet http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=4AA3-0798ENW
[6] HP T300 Data Sheet http://h10088.www1.hp.com/gap/download/products/T300-Color-Inkjet-Web-Press/WebPress_IHPS_DS_US.PDF
[7] HP T350 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T350_US.pdf
[8] HP T400 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T400_US.pdf
[9] HP Indigo w3250 Data Sheet http://ccserver.copiercatalog.com/catalogfiles/HP_Indigo_w3250_sales1.pdf
[10] HP Indigo 5000 Data Sheet http://h10088.www1.hp.com/gap/Data/en/us/5000_DS_Low.pdf
[11] HP Indigo 7500 Data Sheet http://www.csi2.com/resources/HP_Indigo_7500.pdf
[12] PPML Template available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html
[13] PPML Specification v1.5 PDF available at http://www.standards.podi.org/ppml/specification.html

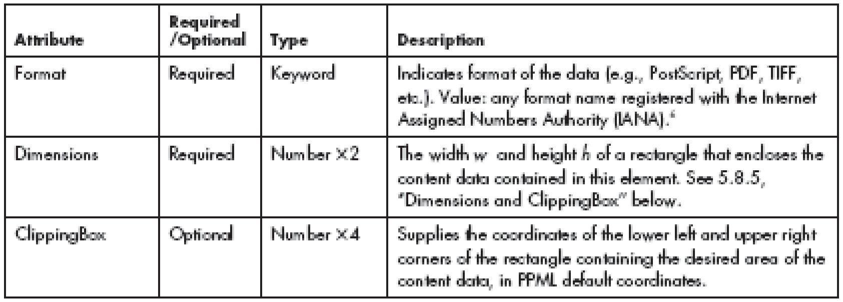
[14] PPML Specification v2.1 PDF available at http://www.standards.podi.org/ppml/specification.html
[15] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf
[16] HP Indigo Yours Truly Designer 7 User Guide (attached)
[17] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.

U.S. Patent No. 5,729,665 ("the '665 patent")

'665 Patent, Claim 1	
1. A method for generating multiple bit maps suitable for high-speed printing or plate-making comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. HP also induces O'Neil's direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the '665 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the '665 patent and of the technology claimed within the '665 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '665 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '665 patent. Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers.

'665 Patent, Claim 1	
	O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. Each of Inkjet Web Presses and Indigo Digital Presses receives print job information from at least one press controller, as further described below.
(a) generating a page description code representing a template, said page description code defining at least one variable data area and said page description code further defining a graphics state corresponding to said variable data area, said graphics state including at least one attribute which controls the appearance of variable data in said variable data area;	O'Neil's OneSuite$^{\text{TM}}$ website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. To the extent that third-parties, such as O'Neil's customers and/or their print media agents, perform the step of generating these files, O'Neil directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. The OneSuite$^{\text{TM}}$ website portal and/or instructions provided through the website portal directs third-parties to provide print specification files such that O'Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, upon information and belief, O'Neil enters contracts with these third parties through which O'Neil enforces the obligations that it imposes upon third-parties. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer.PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and

'665 Patent, Claim 1	
	the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.

<table>
<tr><th>'665 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td><table><tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr><tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr><tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr><tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr></table>Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(b) executing said page description code to generate a bit map of said template, and during said execution, identifying said variable data area defined by said page description code and reserving said graphics state corresponding to said variable data area upon said</td><td>O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a].
O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files; and creates a template bitmap. The template bitmap is composed of</td></tr>
</table>

'665 Patent, Claim 1	
identification;	reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. The VDP file also defines a variable data area by including information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information

'665 Patent, Claim 1	
	remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data.
(c) retrieving variable data;	The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(d) associating said variable data with said graphics state corresponding to said variable data area;	The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field.
(e) applying said graphics state corresponding to said variable data area to said variable data to generate a variable data bit map; and	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an

‘665 Patent, Claim 1

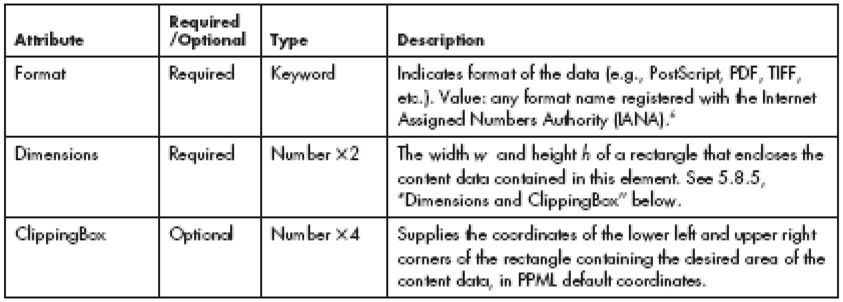
Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [13] at 22; Ref. [14] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

'665 Patent, Claim 1	
	In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.
(f) merging said variable data bit map with said bit map of said template;	The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at 2. Software running on the printer controller interprets PPML, PPMLT, and JLYT files according to the structures defined for each of these VDP files types. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.
wherein said graphics state corresponding to said variable data area is applied repeatedly to variable data to generate a multitude of variable data bit maps without the need to repeat said executing step (b).	The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document.

‘665 Patent, Claim 12	
12. The method of claim 1 wherein said reserving, retrieving, associated, applying, and merging steps are repeated for each variable data area defined by said page description code.	VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT “for-each” command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

‘665 Patent, Claim 13	
13. The method of claim 12 wherein said reserving, retrieving, associating, applying and merging steps are activated by a control task running in a printer controller, and wherein said control task interrupts said page description code execution upon identifying a predetermined command in said page description code.	As mentioned earlier, the steps of reserving, retrieving, associating, applying and merging are all activated and monitored by a control task running in the HP printer controller. On information and belief, the control task interrupts said page description code execution upon identifying a predetermined command in said page description code to enable other operations to be performed.

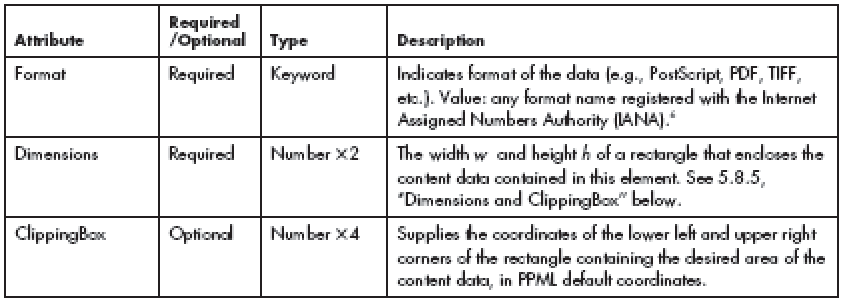
'665 Patent, Claim 20	
20. A method for generating a plurality of bit maps suitable for high-speed printing or plate-making from a page description code representing a template and defining at least one variable data area, and from a merge file containing a plurality of data records of at least one variable data field type, the method comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. HP also induces O'Neil's direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the '665 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the '665 patent and of the technology claimed within the '665 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '665 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '665 patent. Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates Inkjet Web Presses and Indigo Digital Presses supplied by HP. Each of these digital presses receives print job information from at least one press controller, as further described below. O'Neil's OneSuite^{TM} website portal provides O'Neil's products and services to third-party

'665 Patent, Claim 20	
	customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines size and location for static and variable data areas, and further provides appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37.

‘665 Patent, Claim 20	

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Auhority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.

The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable

'665 Patent, Claim 20	
	OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
executing a page description code interpretive program, said interpretive program generates graphics states for each data area defined by said page description code;	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server. . Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including JLYT/SNAP and PPMLT. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which processes PPML, PPMLT, and JLYT files. Ref. [4a]. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an

‘665 Patent, Claim 20	

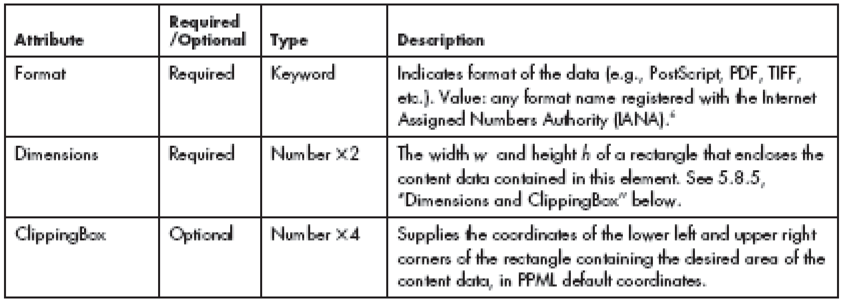
Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [13] at 22; Ref. [14] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



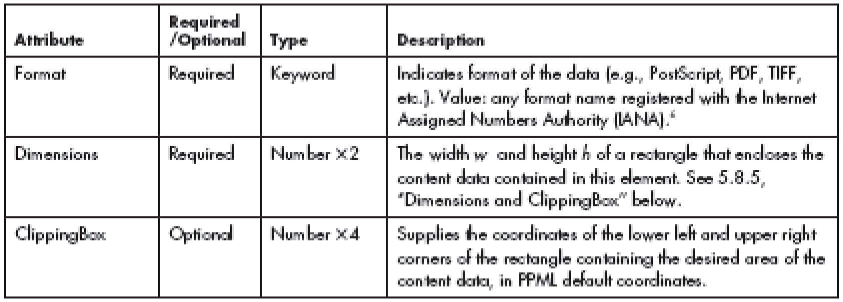
Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

'665 Patent, Claim 20	
	In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.
executing a control task in conjunction with said interpretive program, said control task identifies said variable data area defined by said page description code and reserves said graphics states generated by said interpretive program for said variable data area, said control task generates a template bit map defined by said page description code, and after the completion of said interpretive program, said control task saves said template bit map in memory; and	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server . Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including JLYT/SNAP and PPMLT. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which processes PPML, PPMLT, and JLYT files. Ref. [4a]. HP uses a control task running on these printer controllers to process VDP files and induces O'Neil to do the same, as described above. O'Neil uses such a control task running on these printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files to identify variable data elements by scanning the variable data files and finding the tags associated with such variable data. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)."

'665 Patent, Claim 20	
	Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. The VDP file also defines a variable data area by including information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. The printer controller is also used to create a template bitmap and store a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
executing a merge task upon completion of said interpretive program, said merge task generates variable data bit maps for said data records in said merge file by applying said reserved graphics states to said data records, and said merge task merges said variable data	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content

<table>
<tr><th>'665 Patent, Claim 20</th><th></th></tr>
<tr><td>bit maps with a separate copy of said template bit map to create the plurality of bit maps suitable for high-speed printing or plate making;</td><td>data.
The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34.
"The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36
"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37
"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39.
"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.
<table><tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr><tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr><tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr><tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr></table>Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The</td></tr>
</table>

'665 Patent, Claim 20	
	appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16. The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at 2. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.
whereby said reserved graphics states are applied repeatedly to said data records to generate said variable data bit maps for said data records without the need to repeat said steps of executing a page description code interpretive program and	The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document

'665 Patent, Claim 20	
executing a control task in conjunction with said interpretive program.	instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

U.S. Patent No. 5,937,153 (“the ‘153 patent”)

‘153 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Hewlett-Packard (“HP”), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print (“VDP”) jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O’Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. HP also induces O’Neil’s direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the ‘153 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the ‘153 patent and of the technology claimed within the ‘153 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the ‘153 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the ‘153 patent. Defendant O’Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services to its customers. O’Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O’Neil’s OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O’Neil operates press controllers and presses that process VDP jobs. For example, O’Neil operates Inkjet Web Presses supplied by HPand Indigo Digital Presses supplied by HP. Each of these digital presses receives print job information from at least one press controller, as further described below.
(a) generating a page description code specification,	O’Neil’s OneSuite^{TM} website portal provides O’Neil’s products and services to third-party customers and their print media agents. O’Neil’s OneSuite^{TM} website portal includes multiple

'153 Patent, Claim 1	
the page description code specification defining at least one data area to become variable, and the page description code further defining a graphics state corresponding to the data area, the graphics state including at least one attribute which controls the appearance of data in the data area;	tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. To the extent that third-parties, such as O'Neil's customers and/or their print media agents, perform the step of generating these files, O'Neil directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. The OneSuite^{TM} website portal and/or instructions provided through the website portal directs third-parties to provide print specification files such that O'Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, upon information and belief, O'Neil enters contracts with these third parties through which O'Neil enforces the obligations that it imposes upon third-parties. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the

‘153 Patent, Claim 1	

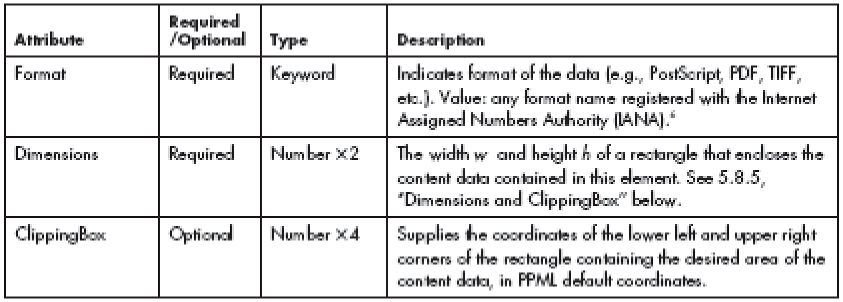
order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [13] at 22; Ref. [14] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is

'153 Patent, Claim 1	
	the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.
(b) interpreting the page description code specification, and during the interpretation, identifying the data area defined by the page description code specification;	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server. . Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. The VDP file defines variable data areas based on the surrounding tags of the data element. Tags within the VDP file include information such as the size and location for each variable data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
(c) storing the graphics state corresponding to the data area upon the identification of the	The VDP file includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of

'153 Patent, Claim 1	
variable data area in step (b);	these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data.
(d) retrieving a variable data item from a plurality of variable data items;	The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(e) applying the stored graphics state to the variable data item to generate a variable data bit map; and	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT,

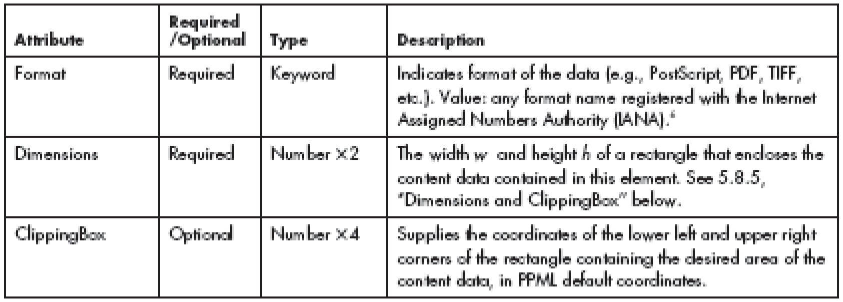
‘153 Patent, Claim 1

REUSABLE_OBJECT and OCCURRENCE.” Ref. [13] at 24; Ref. [14] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number X2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number X4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text

'153 Patent, Claim 1	
	or variable text fields. See Ref. [16] at 16.
(f) repeating steps (d) and (e) for remaining variable data items in the plurality of variable data items, whereby the stored graphics state is applied repeatedly to generate a plurality of variable data bit maps.	The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

'153 Patent, Claim 3	
3. The computer implemented method of claim 1, wherein the page description code specification represents a template and includes a static data area; and the computer implemented method further comprises the steps of:	As described for claim 1 of the '153 patent, O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents a template. These VDP files use static data areas to quickly manage VDP jobs. PPML for example, performs more efficiently when the static data areas are defined in advance. Ref. [12] at 10. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer.

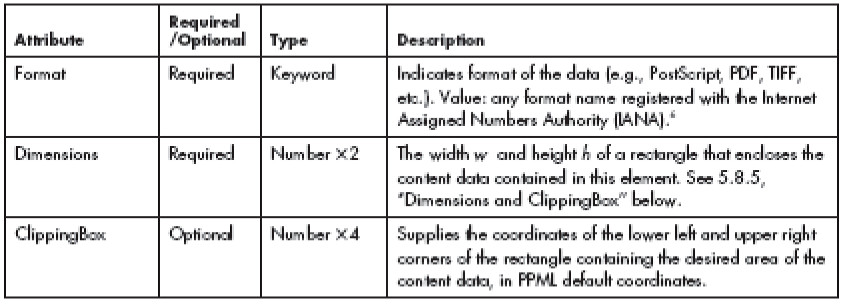
'153 Patent, Claim 3	
executing portions of the page description code specification corresponding to the static data area to generate a template bit map;	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server. . Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
storing the template bit map; and	As described above, the static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
merging each of the plurality of the variable data bit maps into a clean copy of the template bit map to create a plurality of merged bit maps.	The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at 2. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or

'153 Patent, Claim 3	
	environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.

'153 Patent, Claim 4	
4. The computer implemented method of claim 1, wherein the identifying step includes the step of detecting predefined characters within a text string defined in the page description code specification.	As described for claim 1 of the '153 patent, the controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.

'153 Patent, Claim 5	
5. The computer implemented method of claim 1, wherein the attribute is a size attribute, a	As described above, PPML, PPMLT, and JLYT can each define appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, a PPML file includes a hierarchy of elements that define one or more

'153 Patent, Claim 5	
font attribute, a position attribute, an orientation attribute or a location attribute.	jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.

<table>
<tr><th>'153 Patent, Claim 5</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
</table>

'153 Patent, Claim 6	
6. A computer implemented method for processing a page description code specification comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11].

‘153 Patent, Claim 6	
	HP also induces O’Neil’s direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the ‘153 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the ‘153 patent and of the technology claimed within the ‘153 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the ‘153 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the ‘153 patent. Defendant O’Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services to its customers. O’Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O’Neil’s OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O’Neil operates press controllers and presses that process VDP jobs. For example, O’Neil operates Inkjet Web Presses supplied by HPand Indigo Digital Presses supplied by HP. Each of these digital presses receives print job information from at least one press controller, as further described below. O’Neil’s OneSuite^{TM} website portal provides O’Neil’s products and services to third-party customers and their print media agents. O’Neil’s OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O’Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. PPML, PPMLT, and JLYT are standard VDP file types supported by HP’s press controllers and presses such as the ones operated by O’Neil. Refs. [5]-[11].

interpreting the page description code specification, and during the interpretation, identifying a data area defined by the page description code specification;	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
upon the identification of the data area, storing a graphics state set forth in the page	The VDP file defines a variable data area by including information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for

description code specification which defines an attribute of how data is to appear in the data area; and	variable data. Each of the PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data.
repeatedly retrieving data records from a plurality of data records and applying the stored graphics state to the data records to generate a plurality of bitmaps of the data records so that the bitmaps of the data records include the attribute.	The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4. The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field. The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 7; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement

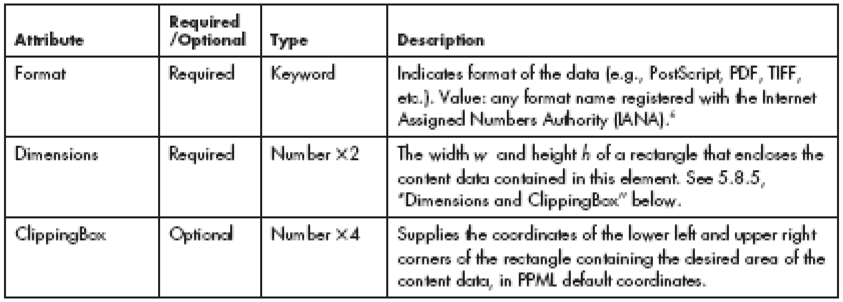
of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34.

"The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36.

"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37.

"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39.

"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

	In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16. The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The printer controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document. VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

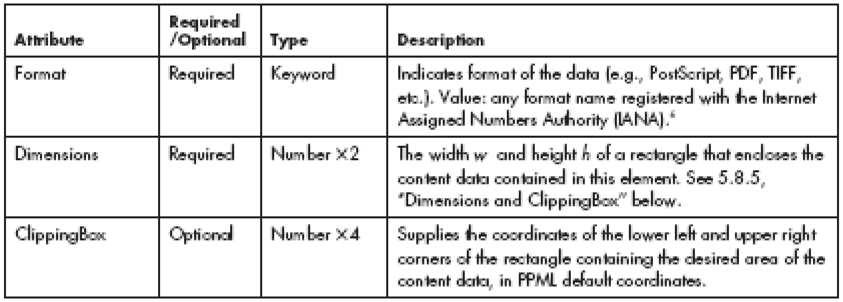
U.S. Patent No. 7,274,479 ("the '479 patent")

'479 Patent, Claim 9	
9. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. HP also induces O'Neil's direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the '479 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the '479 patent and of the technology claimed within the '479 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '479 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '479 patent. Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates Inkjet Web Presses supplied by HPand Indigo digital presses supplied by HP. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below.

'479 Patent, Claim 9	
(a) providing a print specification, the print specification defining at least one variable data area and at least one static data area;	O'Neil's OneSuite$^{\text{TM}}$ website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. Each of these files represents at least one variable data are and at least one static data area. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. Further, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page

'479 Patent, Claim 9	
	objects, for instance)." Ref. [17] at 4-5. JLYT files also include channels that define links to variable content. Ref. [17] at 5.
(b) providing a plurality of variable data items;	The VDP file provides variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(c) identifying the variable data area;	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. The Inkjet Web Presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files. On information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data, as described above.
(d) associating a graphic state with the variable data area, the graphic state including at least one attribute controlling the	The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are

'479 Patent, Claim 9	
appearance of items to be printed in the variable data area;	associated in the file with the "Name" field. Each of the PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36 "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37 "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.

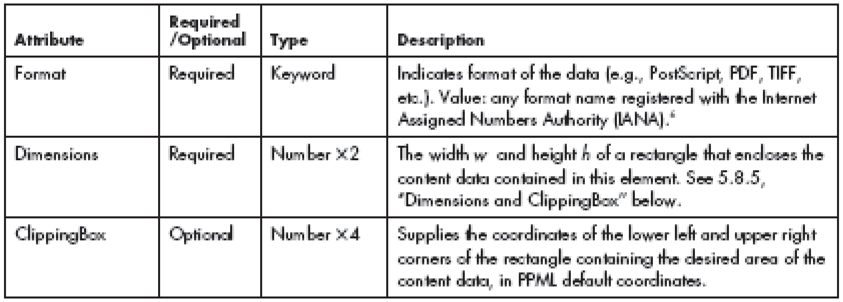
<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>

Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(e) retrieving a variable data item from the plurality of variable data items;</td><td>The printer controller parses the VDP file to access variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24.
Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.</td></tr>
</table>

'479 Patent, Claim 9	
(f) generating a bitmap for the variable item, the generating step including a step of applying the graphic state associated with the variable data area to the variable data item; and	The printer controller and/or the press applies the appearance information to the corresponding variable data to generate a variable data bit map. *See* Ref. [12] at 54; Ref. [15] at 2. The printer controller and/or the press associates the appearance information found in the VDP file to the corresponding variable data that it retrieved from the file. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36 "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [13] at 25; Ref. [14] at 37 "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27; Ref. [14] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [13] at 28; Ref. [14] at 40.



<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [13] at 28; Ref. [14] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(g) repeating steps (e) and (f) for remaining variable data items in the plurality of variable data items, whereby the graphic state associated with the variable data area is applied repeatedly to generate a</td><td>The press controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The press controller creates multiple variable data bitmaps. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the press controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example,</td></tr>
</table>

'479 Patent, Claim 9	
plurality of variable data bitmaps.	uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.

'479 Patent, Claim 10	
10. The method of claim 9, wherein the graphic state associated with the variable data area is defined within the print specification.	Each of the PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data, as discussed above with respect to element (d) of claim 9 of the '479 Patent. The appearance information may be defined within the print specification either by referencing an external file or by providing the appearance information directly within the VDP file.

'479 Patent, Claim 15	
15. The method of claim 9, wherein the variable data area and the static data area are defined, at least in part, by page description language commands.	As described for claim 9 of the '479 Patent, the VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF

'479 Patent, Claim 15	
	elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.

'479 Patent, Claim 17	
17. The method of claim 9, further comprising a step of caching a representation of the static data area, whereby the cached representation of the static data area is available for merging with the variable data bitmaps to generate merged documents.	The static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the printer controller. *See* Ref. [17] at 5.

'479 Patent, Claim 18	
18. The method of claim 17, wherein the cached representation of the static data area is a bitmap representation.	The cached representation of the static data area is a bitmap to avoid the redundant burden of the system to continually compute the contents appearance, as discussed above for claim 17 of the '479 Patent. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7.

'479 Patent, Claim 19	
19. The method of claim 9, wherein: the plurality of data items are associated with a field name; and	As described for claim 9 of the '479 Patent, each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file.
the step of identifying a variable data area includes the step of detecting, in the print specification, a character string associated with the variable data area that matches the field name associated with the plurality of data items.	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file, when contained within a MARK tag denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name

'479 Patent, Claim 19	
	encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.

U.S. Patent No. 7,333,233 (“the ‘233 patent”)

‘233 Patent, Claim 12	
12. A computer implemented method for generating a static bitmap suitable for high-speed variable printing, comprising the steps of:	Defendant Hewlett-Packard (“HP”), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print (“VDP”) jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O’Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. HP also induces O’Neil’s direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the ‘233 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the ‘233 patent and of the technology claimed within the ‘233 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the ‘233 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the ‘233 patent. Defendant O’Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services to its customers. O’Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O’Neil’s OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. In addition to software, O’Neil operates press controllers and presses that process VDP jobs. For example, O’Neil operates Inkjet Web Presses supplied by HPand Indigo Digital Presses supplied by HP Each of these digital presses receives print job information from at least one press controller, as further described below.
providing a page description	O’Neil’s OneSuite$^{\text{TM}}$ website portal provides O’Neil’s products and services to third-party

'233 Patent, Claim 12	
language file, the page description language file defining at least one variable data area and at least one static data area;	customers and their print media agents. O'Neil's OneSuite$^{\text{TM}}$ website portal includes multiple tools used within its VDP process, e.g., ONEdms$^{\text{TM}}$, ONEcard$^{\text{TM}}$, and ONEkit$^{\text{TM}}$. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer.PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines size and location for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12)." Ref. [13] at 22; Ref. [14] at 34. The OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. Thus, PPMLT files provide at least the same size and location information provided by PPML files, because the PPMLT standard fully incorporates the PPML standard. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16,

'233 Patent, Claim 12	
	41-54. In yet another example, JLYT files provide a variety of information to define static and variable data areas. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
interpreting the page description language file, and during the interpreting step, generating a static bitmap of the static data area;	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server.. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files; and creates a template bitmap. On information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer.. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and

'233 Patent, Claim 12	
	redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13.
and saving the static bitmap, whereby the saved static bitmap is used repeatedly in the generation of a plurality of documents, each of which contains the static bitmap and a variable data bitmap.	The static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2.

'233 Patent, Claim 14	
14. A computer implemented method for generating a plurality of bitmaps suitable for high-speed printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. HP also induces O'Neil's direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the '233 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the '233 patent and of the technology claimed within the '233 patent, HP has continued these acts of inducement with

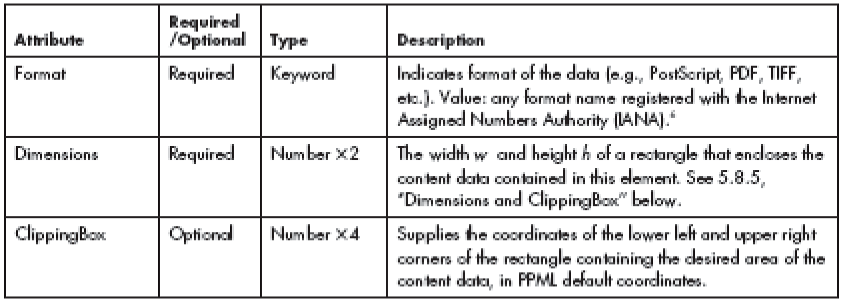
'233 Patent, Claim 14	
	specific intent to cause and/or encourage such direct infringement of the '233 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '233 patent. Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates Inkjet Web Presses supplied by HPand Indigo Digital Presses supplied by HP. Each of these digital presses receives print job information from at least one press controller, as further described below.
(a) providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	O'Neil's OneSuite^{TM} website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files. In addition, on information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. Each of these file types defines size and location for static and variable data. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source

'233 Patent, Claim 14	
	element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12)." Ref. [13] at 22; Ref. [14] at 34. The OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. Thus, PPMLT files provide at least the same size and location information provided by PPML files, because the PPMLT standard fully incorporates the PPML standard. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In yet another example, JLYT files provide a variety of information to define static and variable data areas. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [17] at 4. JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.
(b) providing a merge file including a plurality of variable data items;	HP runs software on a printer controller to parse the VDP files that it generates and/or receives and induces O'Neil to do the same. The VDP files can use variable data elements stored internally or in separate files. In PPML documents, variable data is contained within a non-reusable OBJECT tag, which stores data either internally or externally. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields

'233 Patent, Claim 14	
	internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(c) processing the page description language file, and during the processing step, generating a static bitmap of the static data area and associating the variable data area with the plurality of variable data items; and	O'Neil and HP run software on a printer controller to parse the VDP files that they generate and/or receive. Examples of printer controllers include, but are not limited to, the HP Indigo Production Manager digital front end for its Indigo digital presses, the HP SmartStream Production Pro Print Server, the HP SmartStream Production Plus Print Server, and the HP SmartStream Ultra Print Server. . Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files; and creates a static bitmap. On information and belief, HP also has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. The static bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For

‘233 Patent, Claim 14	
	example, the OBJECT tag within a PPML file “associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to “content packages” that “include any static content in the file (text and image page objects, for instance).” Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. The printer controller and/or the press associates the variable data found in the VDP file to the corresponding variable data area that it retrieved from the file. Each variable data field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, “Name” data in a given record is matched to variable data areas that are associated in the file with the “Name” field. The VDP file provides variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the printer controller. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [12] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., “<xsl: value-of select=’name’/>” points to a database entry for the “name” element. Ref. [12] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. Ref. [17] at 4.
(d) saving the static bitmap;	The static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page

'233 Patent, Claim 14	
	content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2.
(e) generating a first variable data bitmap of a first one of the variable data items utilizing a graphics state associated with the variable data area;	The printer controller and/or the press applies appearance information to the corresponding variable data to generate a variable data bit map. Ref. [12] at 54; Ref. [15] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [13] at 24; Ref. [14] at 36 "The TRANSFORM element represents a two-dimensional homogeneous transformation

<table>
<tr><th>‘233 Patent, Claim 14</th><th></th></tr>
<tr><td></td><td>matrix…TRANSFORM can occur in VIEW.” Ref. [13] at 25; Ref. [14] at 37
“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [13] at 27; Ref. [14] at 39.
“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [13] at 28; Ref. [14] at 40.

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [13] at 28; Ref. [14] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [12] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [16] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [17] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [16] at 16.</td></tr>
<tr><td>(f) merging the first variable</td><td>The printer controller merges the variable data bit map with the template bit map. *See* Ref. [15] at</td></tr>
</table>

'233 Patent, Claim 14	
data bitmap with a copy of the static bitmap to produce a first output bitmap;	2. PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template. For example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [13] at 21; Ref. [14] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [12] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [17] at 4-5.
(g) generating a next variable data bitmap of a next one of the variable data items utilizing a graphics state associated with the variable data area;	The printer controller and/or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. The press controller creates multiple variable data bitmaps, according to the contentions with respect to element (e). Appearance information is reused for each instance of the document. To render each additional variable data record, the printer controller applies the appearance information to each variable data area defined in the VDP file. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [17] at 4.
and (h) merging the next variable data bitmap with a copy of the static bitmap to	The press controller merges the variable data bitmaps with the template bitmap according to the contentions with respect to element (f). The appearance information and the template bitmap are reused for each instance of the document. The template bitmap is only rendered once, while the

'233 Patent, Claim 14	
produce a next output bitmap;	variable data bitmaps must be generated for each variable data area in the subsequent documents. The template bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2.
and (i) repeating steps (g) (h) for remaining variable data items in the plurality of variable data items.	The activities performed for steps (g) and (h) are repeated for each remaining variable data item in the plurality of data items.

Exhibit 4
to Maloney Declaration

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

INDUSTRIAL PRINT TECHNOLOGIES LLC,	)	
	)	
Plaintiff,	)	
	)	
v.	)	Case No. 2:15-CV-00020-JRG
	)	
O'NEIL DATA SYSTEMS, INC., AND HEWLETT-PACKARD COMPANY,	))	
	)	
Defendants.	)	
	)	

PLAINTIFF INDUSTRIAL PRINT TECHNOLOGIES' DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS

In accordance with Patent Local Rules 3-1 and 3-2, plaintiff Industrial Print Technologies LLC ("IPT") submits its Disclosure of Asserted Claims and Infringement Contentions as to Defendants O'Neil Data Systems, Inc. ("O'Neil") and Hewlett-Packard Company ("HP") (collectively, "Defendants").

1. Right to Supplement

IPT bases these disclosures on its current knowledge, understanding and belief as to the facts and information available to it as of the date of these disclosures. This case is not yet in discovery, and IPT has not yet completed its investigation, collection of information, discovery or analysis related to this action. Accordingly, IPT reserves the right to supplement, amend or modify the information contained herein and to use and introduce such information and any subsequently-identified information at trial. In particular, IPT reserves its right to amend and supplement its identification of asserted claims and modify its identification of accused products and instrumentalities. Additionally, as further discovery is taken, and additional details are provided regarding Defendants' activities, IPT's infringement charts and contentions may need to be amended, supplemented and/or modified. IPT also reserves its right to supplement its disclosure of documents based upon further investigation and discovery.

These disclosures are based at least in part upon IPT's present understanding of the meaning and scope of the claims of the patents-in-suit, in the absence of additional claim construction proceedings or discovery. IPT reserves the right to seek leave to supplement or amend these disclosures if its understanding of the claims changes, including if the Court construes them.

2. Asserted Claims

In accordance with Patent LR 3-1(a), based on information presently available to IPT, IPT states that Defendants infringe:

U.S. Patent No. 6,381,028 ("the '028 patent"), claim 4.

IPT reserves the right to assert additional claims against Defendants based upon results of discovery and further investigation.

3. Accused Instrumentalities and Comparison To Asserted Claims

In accordance with Patent LR 3-1(b), based on information presently available to IPT, Defendant O'Neil has been and is engaged in infringing activities using variable data enabled high-speed printing presses supplied by Defendant HP. Specifically, O'Neil is engaged in infringing the asserted method claims through its use of HP's high-speed printing presses that process variable data print jobs, including HP's Inkjet Web Presses (including for example at least T200, T300, T350 and T400 presses) and its Indigo Digital Presses (including for example at least W3250, 3550, WS4600, 5000, 5600, WS6600, WS6600p, W7250, 7500, 7600, 10000, 20000, and 30000 presses).

To the extent that any steps of the methods covered by the asserted patent claims are performed by third-parties, such as O’Neil’s customers and/or their print media agents, Plaintiff alleges that O’Neil is liable for direct infringement because it directs and controls any such third-party steps, including, for example, by dictating the manner by which the third-parties must supply data to enable variable data print jobs to be run on O’Neil’s variable data enabled high-speed printing presses, such that O’Neil is jointly and severally and/or vicariously liable for any acts performed by such third-parties on behalf of O’Neil. Upon information and belief, O’Neil provides an Internet website portal through which it provides its products and services to third-party customers and their print media agents. The website portal and/or instructions provided through the website portal directs these third-parties to provide print specification files such that O’Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, O’Neil enters contracts with these third parties, through which O’Neil enforces the obligations that it imposes upon third-parties.

O’Neil has also induced, and continues to induce, these third parties’ direct infringement of the asserted claims pursuant to pursuant to 35 U.S.C. § 271(b) by providing the Internet website portal through which it provides its products and services to third-party customers and their print media agents, together with instructions directing third-parties’ use of print specification files. Despite its awareness of the asserted claims and of the technology claimed within the asserted claims, O’Neil has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the asserted patent claims and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the asserted patent claims.

HP directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses and Indigo Digital Presses. HP, directly and/or through its subsidiaries, affiliates, agents, and/or business partners has also induced and continues to induce O'Neil's direct infringement of the asserted claims pursuant to 35 U.S.C. § 271(b) by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the asserted claims. HP has also supplied related training and support materials and services. Despite its awareness of the asserted claims and of the technology claimed within the asserted claims, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the asserted patent claims and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the asserted patent claims.

In accordance with Patent LR 3-1(c), IPT provides the following charts, attached as Appendices A and B, which identify specifically where each element and/or step of each asserted claim is found within the Defendants. IPT reserves the right to amend, supplement and modify its contentions and charts based on additional information identified through discovery.

4. Literal and Equivalents Infringement

In accordance with Patent LR 3-1(d), as supported and explained in the attached Exhibits, it is currently believed that each of the elements of each of the asserted claims is met literally, and if any claim or claim limitation is not met literally, then it is met under the doctrine of equivalents.

It is expected that the same facts upon which IPT's literal infringement claim is based will also form the basis of IPT's doctrine of equivalents claim, as any differences between the limitations of the asserted claims and the accused products are insubstantial. With respect to the doctrine of equivalents, however, as Defendants have not yet provided details of their non-infringement positions, IPT reserves the right to present further facts to support an assertion of infringement under the doctrine of equivalents.

5. Priority Date

In accordance with Patent LR 3-1(e), IPT alleges that each asserted claim of all four asserted patents is entitled to a priority date at least as early as January 18, 1995, which is the filing date of U.S. Patent No. 5,729,665, to which the '028 patent claims priority.

The subject matter of the asserted claim of the '028 patent was conceived of prior to the filing of the application that became the '665 patent.

IPT believes that the subject matter of the asserted claim was conceived of at least as early as 1988, and no later than 1989. The subject matter of the asserted claim was then diligently reduced to practice through the first operating prototype that was completed on or about February 10, 1994. IPT thus contends that the asserted claim is entitled to an invention date during 1989. There was constructive reduction to practice on January 18, 1995. To the extent that further investigation and discovery permits a more specific invention date to be confirmed, IPT will update its disclosures as appropriate.

6. Documents

IPT has made a reasonable investigation for documents identified in P.R. 3-2. Such non-privileged documents are being produced herewith**.**

In accordance with Patent LR 3-2, IPT's documents corresponding to P.R. 3-2(a) include at least those numbered:

TES002976-TES002980, TES004201-TES004202, TES004207-TES004209, TES004210-TES004211, TES004212-TES004245, TES004250-TES004278, TES004279-TES004280, TES004281-TES004282, TES004283-TES004284, TES004320-TES004324, TES004325-TES004330, TES004331-TES004333, TES004415-TES004415, TES004416-TES004416, TES004812-TES004812, TES004813-TES004814, TES004822-TES004827, TES004828-TES004833, TES004834-TES004838, TES004843-TES004844, TES004847-TES004848, TES004858-TES004860, TES004861-TES004863, TES004864-TES004866, TES004867-TES004869, TES005505-TES005521, TES005522-TES005527, TES009900-TES010246, TES011201-TES011202, TES013273-TES013304, TES013477-TES013478, TES015782-TES015786, TES018684-TES018720, TES036025-TES036138, TES107224-TES107234, TES108742-TES108775, TES108776-TES108798, TES108799-TES108821, TES237440-TES237442, TES240475-TES240608.

IPT's documents corresponding to P.R. 3-2(b) include at least those numbered:

TES002250-TES002253, TES002269-TES002271, TES002305-TES002422, TES002870-TES002873, TES003038-TES003047, TES003856-TES003993, TES003998-TES004029, TES004077-TES004078, TES004083-TES004100, TES004104-TES004104, TES004119-TES004163, TES004184-TES004197, TES004203-TES004203, TES004207-TES004245, TES004250- ES004284, TES004286-TES004291, TES004293-TES004303, TES004305-TES004310, TES004320-TES004333, TES004365-TES004398, TES004403-TES004405, TES004409-TES004409, TES004417-TES004420, TES004445-TES004455, TES004478-TES004478, TES004480-TES004480, TES004481-TES004487, TES004489-TES004523, TES004525-TES004545, TES004551-TES004551, TES004579-TES004607, TES004614-TES004665, TES004669- TES004717, TES004724-TES004729, TES004731-TES004798, TES004812-TES004814, TES004816-TES004871, TES004880-TES005020, TES005028-TES005099, TES005107-TES005290, TES005299-TES005304, TES005306-TES005350, TES005352-TES005380, TES005382-TES005383, TES005385-TES005437, TES005457-TES005458, TES005460-TES005470, TES005505-TES005521, TES005528, TES005532-TES005540, TES005564-TES005591, TES005598-TES005607, TES005609-TES005645, TES005672-TES005680, TES006504-TES006653, TES006695-TES006695, TES006723-TES006724, TES006816-TES006846, TES007208-TES007223, TES008359-TES008448, TES008463-TES008584, TES008614-TES008620, TES008650-TES008680, TES008691-TES008695, TES009442-TES009503, TES009525-TES009537, TES009595, TES009659-TES009662, TES009848-TES009899, TES011141-TES011200, TES011203-TES011303, TES011310-TES011372, TES011608-TES011669, TES011817-TES011820, TES011823-TES011986, TES012004-TES012014, TES012040-TES012054, TES012290-TES012354, TES013081-TES013174, TES013273-TES013304, TES014021-TES014151, TES014190-TES015304, TES015787-TES015799, TES015810-TES015813, TES016292-TES016334, TES018613-TES018623, TES018626-TES018679, TES019295-TES019351, TES019356-TES019379, TES022843-TES022853, TES023472-TES023476, TES025611-TES025624, TES025626-TES025679, TES032626-TES032657, TES032664-TES032695, TES038176-TES038282, TES038419-TES038585, TES038623-TES038694, TES038829-

TES039181, TES040237-TES040526, TES040784-TES041088, TES041343-TES041422, TES047510-TES047514, TES100247-TES100251, TES100286-TES100287, TES100293-TES100326, TES100580-TES100580, TES100604-TES100610, TES107224-TES107234, TES274326-TES274326, TES279177-TES279177, TES280365-TES280365, TES280374-TES280374, TES281386-TES281386, TES281730-TES281730, TES281739-TES281739, TES281747-TES281747.

IPT's documents corresponding to P.R. 3-2(c) are numbered:

TES336688-TES336813, TES337205-TES337279, TES337507-TES337622, TES337623-TES337713, TES338116-TES338285, TES338286-TES338324, TES340745-TES342864, TES342865-TES344969, TES344970-TES347044, TES347045-TES349151, TES349455-TES352270, TES352271-TES355288.

Date: February 11, 2015

/s/ Alison Aubry Richards
Timothy P. Maloney (IL 6216483)
Alison Aubry Richards (IL 6285669)
Nicole L. Little (IL 6297047)
David A. Gosse (IL 6299892)
FITCH, EVEN, TABIN & FLANNERY
120 South LaSalle Street, Suite 1600
Chicago, Illinois 60603
Telephone: (312) 577-7000
Facsimile: (312) 577-7007

T. John Ward, Jr.
Texas State Bar No. 00794818
Email: jw@wsfirm.com
Wesley Hill
Texas State Bar No. 24032294
Email: wh@wsfirm.com
WARD & SMITH
Post Office Box 1231
Longview, TX 75606
Telephone: (903) 757-6400
Facsimile: (903) 757-2323

Steven C. Schroer (IL 6250991)
scschr@fitcheven.com
FITCH, EVEN, TABIN & FLANNERY
1942 Broadway
Suite 213
Boulder, CO 80302
Telephone: 303.402.6966
Facsimile: 303.402.6970

Counsel for Plaintiff

CERTIFICATE OF SERVICE

The undersigned certifies that a copy of the above document PLAINTIFF IPT'S DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS and exhibits was sent by email and first class mail to the counsel listed below on this February 11, 2015:

Edward R. Reines
edward.reines@weil.com
Sonal N. Mehta
sonal.mehta@weil.com
Evan N. Budaj
evan.budaj@weil.com
WEIL, GOTSHAL & MANGES LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

Audrey L. Maness
WEIL, GOTSHAL & MANGES LLP
700 Louisiana, Suite 1700
Houston, TX 77002
Telephone: (713) 546-5317
Facsimile: (713) 224-9511

Melissa Richards Smith
melissa@gillamsmithlaw.com
GILLAM & SMITH, LLP
303 South Washington Avenue
Marshall, TX 75670
Telephone: (903) 934-8450
Facsimile: (903) 934-9257

/s/ Alison Aubry Richards
Alison Aubry Richards
Attorney for Plaintiff

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. Discovery in this case has not yet commenced, and the charts below do not reflect any information produced by defendants O'Neil Data Systems, Inc. or Hewlett-Packard Company. IPT reserves the right to support its theories with additional material produced by the defendants or subsequently identified by IPT.

[1] O'Neil Data Solutions website http://www.oneildata.com/services/onesuite/onedms
[2] HP, O'Neil Data Systems: HP Indigo Presses Power Targeted Marketing Campaigns, available at http://h10088.www1.hp.com/gap/download/O_Neil_Data_Systems_HP_Indigo_presses_Case_Study.pdf
[3] O'Neil Data Systems and the HP T400 Spearhead Industry Change, available at http://www.oneildata.com/hp-large-format-printing/oneil-data-systems-and-the-hp-t400-spearhead-industry-change/
[4] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf
[4a] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf
[5] HP T200 Data Sheet http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=4AA3-0798ENW
[6] HP T300 Data Sheet http://h10088.www1.hp.com/gap/download/products/T300-Color-Inkjet-Web-Press/WebPress_IHPS_DS_US.PDF
[7] HP T350 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T350_US.pdf
[8] HP T400 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T400_US.pdf
[9] HP Indigo w3250 Data Sheet http://ccserver.copiercatalog.com/catalogfiles/HP_Indigo_w3250_sales1.pdf
[10] HP Indigo 5000 Data Sheet http://h10088.www1.hp.com/gap/Data/en/us/5000_DS_Low.pdf
[11] HP Indigo 7500 Data Sheet http://www.csi2.com/resources/HP_Indigo_7500.pdf
[12] PPML Template available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html
[13] PPML Specification v1.5 PDF available at http://www.standards.podi.org/ppml/specification.html

[14] PPML Specification v2.1 PDF available at http://www.standards.podi.org/ppml/specification.html
[15] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf
[16] HP Indigo Yours Truly Designer 7 User Guide (attached)
[17] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.

U.S. Patent No. 6,381,028 ("the '028 patent")

'028 Patent, Claim 4	
4. A computer implemented method for generating a reusable template bit map suitable for high-speed variable printing, comprising the steps of:	Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers. O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below.
generating a page description code specification, the page description code specification defining at least one variable data area and at least one static data area;	O'Neil's OneSuite^{TM} website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files represents at least one variable data area and at least one

'028 Patent, Claim 4	
	static data area. To the extent that third-parties, such as O'Neil's customers and/or their print media agents, perform the step of generating these files, O'Neil directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. The OneSuite^{TM} website portal and/or instructions provided through the website portal directs third-parties to provide print specification files such that O'Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, upon information and belief, O'Neil enters contracts with these third parties through which O'Neil enforces the obligations that it imposes upon third-parties. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. Further, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref.

'028 Patent, Claim 4	
	[12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files also include channels that define links to variable content. Ref. [17] at 5. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description code specification, and during the interpreting step,	O'Neil runs software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil uses such printer controllers to process VDP files including one or more of PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects.
generating a bitmap of the static data area and adding the bitmap of the static data area to a template bitmap;	O'Neil uses the printer controllers described above to create a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins.

'028 Patent, Claim 4	
	PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the printer controller. *See* Ref. [17] at 5. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
identifying the variable data area, and	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5.

'028 Patent, Claim 4	
	O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
responsive to the identification of the variable data, not adding a bitmap of the variable data area to the template bitmap; and	The static bitmap is reused in subsequent Pages, Documents, Jobs, and Datasets, and therefore does not include a bitmap of the variable data area. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In yet another example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are later stitched together to generate a merged document bitmap. *See* Ref. [15] at 2. In a further example, JLYT files are rasterized to the proprietary Indigo Compressed Format (ICF) and later assembled at the printer controller according to the instructions in the JLYT file. Ref. [17] at 5. Thus, the template bitmap does not include variable data. *See id*. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
saving the template bitmap, whereby copies of the template bitmap can be continuously accessed to create a plurality of variable data bitmaps.	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. VDP files are optimized for handling variable data associated with a series of documents. As

'028 Patent, Claim 4	
	described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller repeatedly renders the variable data area. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records that are merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to render each piece of variable data and merge it with the static bitmap. Ref. [17] at 4. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. Discovery in this case has not yet commenced, and the charts below do not reflect any information produced by defendants O'Neil Data Systems, Inc. or Hewlett-Packard Company. IPT reserves the right to support its theories with additional material produced by the defendants or subsequently identified by IPT.

[1] O'Neil Data Solutions website http://www.oneildata.com/services/onesuite/onedms
[2] HP, O'Neil Data Systems: HP Indigo Presses Power Targeted Marketing Campaigns, available at http://h10088.www1.hp.com/gap/download/O_Neil_Data_Systems_HP_Indigo_presses_Case_Study.pdf
[3] O'Neil Data Systems and the HP T400 Spearhead Industry Change, available at http://www.oneildata.com/hp-large-format-printing/oneil-data-systems-and-the-hp-t400-spearhead-industry-change/
[4] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf
[4a] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf
[5] HP T200 Data Sheet http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=4AA3-0798ENW
[6] HP T300 Data Sheet http://h10088.www1.hp.com/gap/download/products/T300-Color-Inkjet-Web-Press/WebPress_IHPS_DS_US.PDF
[7] HP T350 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T350_US.pdf
[8] HP T400 Data Sheet http://h10088.www1.hp.com/gap/download/HP_Inkjet_Color_Web_Pres_T400_US.pdf
[9] HP Indigo w3250 Data Sheet http://ccserver.copiercatalog.com/catalogfiles/HP_Indigo_w3250_sales1.pdf
[10] HP Indigo 5000 Data Sheet http://h10088.www1.hp.com/gap/Data/en/us/5000_DS_Low.pdf
[11] HP Indigo 7500 Data Sheet http://www.csi2.com/resources/HP_Indigo_7500.pdf
[12] PPML Template available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html
[13] PPML Specification v1.5 PDF available at http://www.standards.podi.org/ppml/specification.html

[14] PPML Specification v2.1 PDF available at http://www.standards.podi.org/ppml/specification.html
[15] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf
[16] HP Indigo Yours Truly Designer 7 User Guide (attached)
[17] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.

U.S. Patent No. 6,381,028 ("the '028 patent")

'028 Patent, Claim 4	
4. A computer implemented method for generating a reusable template bit map suitable for high-speed variable printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at O'Neil facilities, including by operating Inkjet Web Presses supplied by HP, including: HP T200, T300, T350, and T400; and Indigo Digital Presses supplied by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. HP also induces O'Neil's direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses, which were designed and intended to practice methods covered by the '665 patent, and, on information and belief, HP has supplied related training and support materials and services. Despite its awareness of the '665 patent and of the technology claimed within the '665 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '665 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '665 patent. Defendant O'Neil, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services to its customers.

'028 Patent, Claim 4	
	O'Neil provides Internet-based software to its clients, which uses VDP technology to quickly create and print documents containing variable data. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. In addition to software, O'Neil operates press controllers and presses that process VDP jobs. For example, O'Neil operates inkjet web presses manufactured by HP, including: HP T200, T300, T350, and T400; and Indigo digital presses manufactured by HP, including: w3250, 5000, and 7500. Refs. [2]-[11]. Each of these digital presses receives print job information from at least one press controller, as further described below.
generating a page description code specification, the page description code specification defining at least one variable data area and at least one static data area;	On information and belief, HP has software tools that are part of a process by which HP generates, references, and/or incorporates VDP files such as PPML, PPMLT, and JLYT files, including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. PPML, PPMLT, and JLYT are standard VDP file types supported by HP's press controllers and presses such as the ones operated by O'Neil. Refs. [5]-[11]. In addition, O'Neil's OneSuite^{TM} website portal provides O'Neil's products and services to third-party customers and their print media agents. O'Neil's OneSuite^{TM} website portal includes multiple tools used within its VDP process, e.g., ONEdms^{TM}, ONEcard^{TM}, and ONEkit^{TM}. Ref. [1]. These tools are part of a process by which O'Neil generates, references, and/or incorporates VDP files such as PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files represents at least one variable data are and at least one static data area. To the extent that third-parties, such as O'Neil's customers and/or their print media agents, perform the step of generating these files, O'Neil directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. The OneSuite^{TM} website portal and/or instructions provided through the website portal directs third-parties to provide print specification files such that O'Neil can process variable data print jobs according to the remaining steps of the patented invention. Further, upon information and belief, O'Neil enters contracts with these third parties through which O'Neil enforces the obligations that it imposes upon third-parties.

'028 Patent, Claim 4	
	The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file. For example, a PPML file includes a hierarchy of elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects which represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be marked. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [13] at 22; Ref. [14] at 34. Further, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files also include channels that define links to variable content. Ref. [17] at 5. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description code specification, and during	O'Neil and HP run software on a printer controller to parse the VDP files that it generates and/or receives. Among other such printer controllers, O'Neil operates an HP Indigo Production

'028 Patent, Claim 4	
the interpreting step,	Manager digital front end for its Indigo digital presses. Ref. [2]. The HP Indigo Production Manager supports multiple VDP file types including PPML, PPMLT, and JLYT/SNAP. Ref. [4]. O'Neil's inkjet web presses are designed to interface with HP's SmartStream Ultra Print Server, which also processes PPML, PPMLT, and JLYT files. Ref. [4a]. O'Neil and HP use such printer controllers to process VDP files including one or more of PPML, PPMLT, and JLYT files, and/or other VDP file types that are substantially similar in relevant respects.
generating a bitmap of the static data area and adding the bitmap of the static data area to a template bitmap;	O'Neil and HP uses the printer controllers described above to create a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [13] at 11; Ref. [14] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [15] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the printer

'028 Patent, Claim 4	
	controller. *See* Ref. [17] at 5. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
identifying the variable data area, and	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [13] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [13] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
responsive to the identification of the variable data, not adding a bitmap of the variable data area to the template bitmap; and	The static bitmap is reused in subsequent Pages, Documents, Jobs, and Datasets, and therefore does not include a bitmap of the variable data area. For example, with respect to PPML documents, "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by

'028 Patent, Claim 4	
	avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. In yet another example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [12] at 7. The static bitmap and the variable data bitmap are later stitched together to generate a merged document bitmap. *See* Ref. [15] at 2. In a further example, JLYT files are rasterized to the proprietary Indigo Compressed Format (ICF) and later assembled at the printer controller according to the instructions in the JLYT file. Ref. [17] at 5. Thus, the template bitmap does not include variable data. *See id.* O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
saving the template bitmap, whereby copies of the template bitmap can be continuously accessed to create a plurality of variable data bitmaps.	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [13] at 11; Ref. [14] at 13. VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the printer controller repeatedly renders the variable data area. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records that are merged with the static bitmap. Ref. [13] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a

'028 Patent, Claim 4	
	PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [12] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the printer controller. On information and belief, processing the external variable data causes the printer controller to render each piece of variable data and merge it with the static bitmap. Ref. [17] at 4. O'Neil may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

Exhibit 5
to Maloney Declaration

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

Industrial Print Technologies, LLC, Plaintiff, v. Cenveo, Inc. and Hewlett-Packard Company, Defendants.	Civil Action No. 3:15-cv-00165-M The Honorable Barbara M.G. Lynn THIS DOCUMENT RELATES TO CIVIL ACTION NO. 3:15-cv-00165-M

PLAINTIFF INDUSTRIAL PRINT TECHNOLOGIES' DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS

In accordance with Miscellaneous Order No. 62 ¶¶ 3-1 and 3-2, and the Court's Order during the April 20, 2015 telephonic hearing, plaintiff Industrial Print Technologies LLC ("IPT") submits its Disclosure of Asserted Claims and Infringement Contentions as to Defendants Cenveo, Inc. ("Cenveo") and Hewlett-Packard Company ("HP") (collectively, "Defendants").

1. Right to Supplement

IPT bases these disclosures on its current knowledge, understanding and belief as to the facts and information available to it as of the date of these disclosures. Discovery in this case has not yet opened, and IPT has not yet completed its investigation, collection of information, discovery or analysis related to this action. Accordingly, IPT reserves the right to supplement, amend or modify the information contained herein and to use and introduce such information and any subsequently-identified information at trial. In particular, IPT reserves its right to amend and supplement its identification of asserted claims and modify its identification of accused products and instrumentalities. Additionally, as further discovery is taken, and additional details are provided regarding Defendants' activities, IPT may seek to amend and/or supplement. IPT also reserves its right to supplement its disclosure of documents based upon further investigation and discovery.

These disclosures are based at least in part upon IPT's present understanding of the meaning and scope of the claims of the patents-in-suit, in the absence of additional claim construction proceedings or discovery. IPT reserves the right to seek leave to supplement or amend these disclosures if its understanding of the claims changes, including if the Court construes them.

2. Asserted Claims

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(1), based on information presently available to IPT, IPT states that Defendants infringe:

U.S. Patent No. 5,729,665 ("the '665 patent"), claims 1, 12, 13, and 20;

U.S. Patent No. 5,937,153 ("the '153 patent"), claims 1, 3-5, and 6;

U.S. Patent No. 6,381,028 ("the '028 patent"), claims 1, 2, and 4;

U.S. Patent No. 7,274,479 ("the '479 patent"), claims 9, 10, 15, and 17-19; and

U.S. Patent No. 7,333,233 ("the '233 patent"), claims 12 and 14.

IPT reserves the right to assert additional claims against Defendants based upon results of discovery and further investigation.

3. Accused Instrumentalities and Comparison To Asserted Claims

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(2), based on information presently available to IPT, Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has been and is engaged in infringing activities using variable data enabled high-speed printing presses supplied by Defendant HP. Specifically, Cenveo has been and is engaged in infringing the asserted method claims under 35 U.S.C. § 271(a) through its use of HP's high-speed printing presses that process variable data print ("VDP") jobs, including at least HP Indigo Digital Presses (including for example at least Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200 presses) and by selling and/or offering to sell related variable data printing services to its customers within the United States. Cenveo has also been and is infringing under 35 U.S.C. § 271(g) by selling and/or offering to sell print materials containing variable data which are made using methods covered by the patented methods to its customers within the United States.

To the extent that any steps of the methods covered by the asserted patent claims are performed by third-parties, such as Cenveo's customers and/or their print media agents, Plaintiff alleges that Cenveo is liable for direct infringement because it directs and controls any such third-party steps, including, for example, by dictating the manner by which the third-parties must supply data to enable variable data print jobs to be run on Cenveo's variable data enabled high-speed printing presses, such that Cenveo is jointly and severally and/or vicariously liable for any acts performed by such third-parties on behalf of Cenveo. Further, upon information and belief, Cenveo enters contracts with these third parties, through which Cenveo enforces the obligations that it imposes upon third-parties.

HP directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe under 35 U.S.C. § 271(a) by setting up and running VDP jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo facilities, including at least by operating Indigo Digital Presses. HP, directly and/or through its subsidiaries, affiliates, agents, and/or business partners has also induced and continues to induce Cenveo and other HP customers to commit direct infringement of the asserted claims pursuant to 35 U.S.C. § 271(b) by one or more of supplying, offering for sale and selling at least its Indigo Digital Presses, which were designed and intended to practice methods covered by the asserted claims. HP has also supplied related training and support materials and services. Despite its awareness of the asserted claims and of the technology claimed within the asserted claims, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the asserted patent claims and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the asserted patent claims.

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(3), IPT provides the following charts, attached as Appendices A and B, that identify specifically where each element and/or step of each asserted claim is found within the Defendants' infringing methods and systems. IPT reserves the right to amend, supplement and modify its contentions and charts based on additional information identified through discovery.

4. Literal and Equivalents Infringement

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(4), as supported and explained in the attached Appendices, IPT currently believes that each of the elements of each of the asserted claims is met literally, and if any claim or claim limitation is not met literally, then it is met under the doctrine of equivalents.

It is expected that the same facts upon which IPT's literal infringement claim is based will also form the basis of IPT's doctrine of equivalents claim, as any differences between the limitations of the asserted claims and the accused products are insubstantial. With respect to the doctrine of equivalents, however, as Defendants have not yet provided details of their non-infringement positions, IPT reserves the right to present further facts to support an assertion of infringement under the doctrine of equivalents.

5. Priority Date

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(5), IPT alleges that each asserted claim of all five asserted patents is entitled to a priority date at least as early as January 18, 1995, which is the filing date of the ‘665 patent, to which the other asserted patents also claim priority.

The subject matter of the asserted claims of the asserted patents was conceived of prior to the filing of the application that became the ‘665 patent.

IPT believes that the subject matter of the asserted claims was conceived of at least as early as 1988, and no later than 1989. The subject matter of the asserted claims was then diligently reduced to practice through the first operating prototype that was completed on or about February 10, 1994. IPT thus contends that the claims are entitled to an invention date during 1989. There was constructive reduction to practice on January 18, 1995. To the extent that further investigation and discovery permits a more specific invention date to be confirmed, IPT will update its disclosures as appropriate.

6. Documents

IPT has made a reasonable investigation for documents identified in Miscellaneous Order No. 62 ¶ 3-2. Such non-privileged documents are being produced herewith**.**

In accordance with Miscellaneous Order No. 62 ¶ 3-2(a), IPT’s documents corresponding to ¶ 3-2(a)(1) include at least those numbered:

TES002976-TES002980, TES004201-TES004202, TES004207-TES004209, TES004210-TES004211, TES004212-TES004245, TES004250-TES004278, TES004279-TES004280, TES004281-TES004282, TES004283-TES004284, TES004320-TES004324, TES004325-TES004330, TES004331-TES004333, TES004415-TES004415, TES004416-TES004416, TES004812-TES004812, TES004813-TES004814, TES004822-TES004827, TES004828-TES004833, TES004834-TES004838, TES004843-TES004844, TES004847-TES004848, TES004858-TES004860, TES004861-TES004863, TES004864-TES004866, TES004867-TES004869, TES005505-TES005521, TES005522-TES005527, TES009900-

TES010246, TES011201-TES011202, TES013273-TES013304, TES013477-TES013478, TES015782-TES015786, TES018684-TES018720, TES036025-TES036138, TES107224-TES107234, TES108742-TES108775, TES108776-TES108798, TES108799-TES108821, TES237440-TES237442, TES240475-TES240608.

IPT's documents corresponding to ¶ 3-2(a)(2) include at least those numbered:

TES002250-TES002253, TES002269-TES002271, TES002305-TES002422, TES002870-TES002873, TES003038-TES003047, TES003856-TES003993, TES003998-TES004029, TES004077-TES004078, TES004083-TES004100, TES004104-TES004104, TES004119-TES004163, TES004184-TES004197, TES004203-TES004203, TES004207-TES004245, TES004250- ES004284, TES004286-TES004291, TES004293-TES004303, TES004305-TES004310, TES004320-TES004333, TES004365-TES004398, TES004403-TES004405, TES004409-TES004409, TES004417-TES004420, TES004445-TES004455, TES004478-TES004478, TES004480-TES004480, TES004481-TES004487, TES004489-TES004523, TES004525-TES004545, TES004551-TES004551, TES004579-TES004607, TES004614-TES004665, TES004669- TES004717, TES004724-TES004729, TES004731-TES004798, TES004812-TES004814, TES004816-TES004871, TES004880-TES005020, TES005028-TES005099, TES005107-TES005290, TES005299-TES005304, TES005306-TES005350, TES005352-TES005380, TES005382-TES005383, TES005385-TES005437, TES005457-TES005458, TES005460-TES005470, TES005505-TES005521, TES005528, TES005532-TES005540, TES005564-TES005591, TES005598-TES005607, TES005609-TES005645, TES005672-TES005680, TES006504-TES006653, TES006695-TES006695, TES006723-TES006724, TES006816-TES006846, TES007208-TES007223, TES008359-TES008448, TES008463-TES008584, TES008614-TES008620, TES008650-TES008680, TES008691-TES008695, TES009442-TES009503, TES009525-TES009537, TES009595, TES009659-TES009662, TES009848-TES009899, TES011141-TES011200, TES011203-TES011303, TES011310-TES011372, TES011608-TES011669, TES011817-TES011820, TES011823-TES011986, TES012004-TES012014, TES012040-TES012054, TES012290-TES012354, TES013081-TES013174, TES013273-TES013304, TES014021-TES014151, TES014190-TES015304, TES015787-TES015799, TES015810-TES015813, TES016292-TES016334, TES018613-TES018623, TES018626-TES018679, TES019295-TES019351, TES019356-TES019379, TES022843-TES022853, TES023472-TES023476, TES025611-TES025624, TES025626-TES025679, TES032626-TES032657, TES032664-TES032695, TES038176-TES038282, TES038419-TES038585, TES038623-TES038694, TES038829-TES039181, TES040237-TES040526, TES040784-TES041088, TES041343-TES041422, TES047510-TES047514, TES100247-TES100251, TES100286-TES100287, TES100293-TES100326, TES100580-TES100580, TES100604-TES100610, TES107224-TES107234, TES274326-TES274326, TES279177-TES279177, TES280365-TES280365, TES280374-TES280374, TES281386-TES281386, TES281730-TES281730, TES281739-TES281739, TES281747-TES281747.

IPT's documents corresponding to ¶ 3-2(a)(3) include at least those numbered:

TES336688-TES336813, TES337205-TES337279, TES337507-TES337622, TES337623-TES337713, TES338116-TES338285, TES338286-TES338324, TES340745-

TES342864, TES342865-TES344969, TES344970-TES347044, TES347045-TES349151, TES349455-TES352270, TES352271-TES355288.

Date: May 11, 2015

/s/ David A. Gosse
Timothy P. Maloney (IL 6216483)
Alison Aubry Richards (IL 6285669)
Nicole L. Little (IL 6297047)
David A. Gosse (IL 6299892)
FITCH, EVEN, TABIN & FLANNERY LLP
120 South LaSalle Street, Suite 1600
Chicago, Illinois 60603
Telephone: (312) 577-7000
Facsimile: (312) 577-7007

Steven C. Schroer (IL 6250991)
scschr@fitcheven.com
FITCH, EVEN, TABIN & FLANNERY LLP
1942 Broadway, Suite 213
Boulder, CO 80302
Telephone: 303.402.6966
Facsimile: 303.402.6970

Counsel for Plaintiff

CERTIFICATE OF SERVICE

The undersigned certifies that a copy of the above document PLAINTIFF IPT'S DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS and exhibits was sent by email to the counsel listed below on this May 11, 2015:

Edward R. Reines
edward.reines@weil.com
Sonal N. Mehta
sonal.mehta@weil.com
Evan N. Budaj
evan.budaj@weil.com
WEIL, GOTSHAL & MANGES LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

Audrey L. Maness
WEIL, GOTSHAL & MANGES LLP
700 Louisiana, Suite 1700
Houston, TX 77002
Telephone: (713) 546-5317
Facsimile: (713) 224-9511

Melissa Richards Smith
melissa@gillamsmithlaw.com
GILLAM & SMITH, LLP
303 South Washington Avenue
Marshall, TX 75670
Telephone: (903) 934-8450
Facsimile: (903) 934-9257

Counsel for Defendants Cenveo, Inc. and Hewlett-Packard Co.

/s/ David A. Gosse
David A. Gosse
Attorney for Plaintiff

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. These contentions are based solely on publicly available information relating to exemplary variable data ("VDP") files and raster image processor (RIP") software and press control software used on HP's presses. Discovery in this case has not yet begun, and the charts below do not reflect any information produced by defendants Cenveo, Inc. and Hewlett-Packard Company. Other VDP files, RIP software and press control software may be identified through discovery as being used by defendants. IPT reserves the right to support its contentions with additional material produced by the defendants or subsequently identified by IPT.

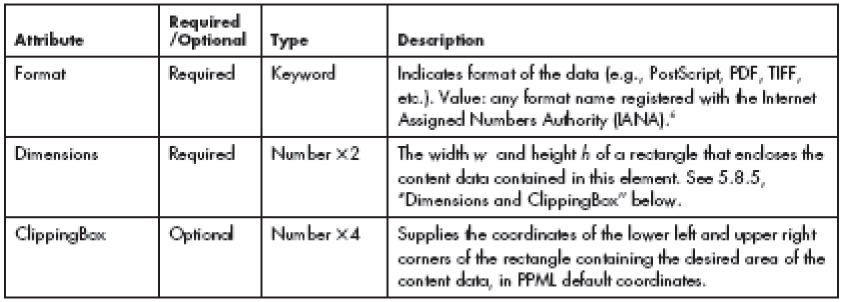
[1] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf
[2] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf
[3] HP T200 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T200_Data_Sheet.pdf
[4] HP T300 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T300_Data_Sheet.pdf
[5] HP T350 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T350_Data_Sheet.pdf
[6] HP T400 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T400_Data_Sheet.pdf
[7] HP Indigo w3250 Data Sheet, available at http://h10010.www1.hp.com/wwpc/pscmisc/vac/us/product_pdfs/90566.pdf
[8] HP Indigo 5600 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2012/HPPredrupa12/HP_Indigo_5600.pdf
[9] HP Indigo 7500 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2010/IPEX2010/HP_Indigo_7500_DS.PDF
[10] PPML Template, available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html
[11] PPML Specification v1.5, available at http://www.standards.podi.org/ppml/specification.html
[12] PPML Specification v2.1, available at http://www.standards.podi.org/ppml/specification.html

[13] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf
[14] HP Indigo Yours Truly Designer 7 User Guide
[15] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.
[16] Adobe Systems Inc., PDF Reference 5th Ed., v. 1.6, available at http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference16.pdf
[17] ISO 16612-2:2010, available at http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=46428
[18] Global Graphics, Do PDF/VT Right, available at http://www.globalgraphics.com/doPDFVTright/

U.S. Patent No. 5,729,665 ("the '665 patent")

'665 Patent, Claim 1	
1. A method for generating multiple bit maps suitable for high-speed printing or plate-making comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code representing a template, said page description code defining at least one	Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in element (b) below. Each of these files further defines at least one

'665 Patent, Claim 1	
variable data area and said page description code further defining a graphics state corresponding to said variable data area, said graphics state including at least one attribute which controls the appearance of variable data in said variable data area;	variable data area, as described further in element (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. To the extent that third-parties, such as Cenveo's customers and/or their print media agents, perform the step of generating these files, Cenveo directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo enters contracts with these third parties through which Cenveo enforces the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data.

<table>
<tr><th>'665 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>
<p>The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34.</p>
<p>"The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.</p>
<p>"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.</p>
<p>"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.</p>
<p>"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.</p>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
</td></tr>
</table>

'665 Patent, Claim 1	
	In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) executing said page description code to generate a bit map of said template, and during said execution, identifying said variable data area defined by said page description code and reserving said graphics state corresponding to said variable data area upon said identification;	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5.

'665 Patent, Claim 1	
	For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By

'665 Patent, Claim 1	
	interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it

'665 Patent, Claim 1	
	can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) retrieving variable data;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions

'665 Patent, Claim 1	
	similar to those described above in these exemplary file types.
(d) associating said variable data with said graphics state corresponding to said variable data area;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to associate the appearance information found in the VDP file to the corresponding variable data. As described above, variable data may be stored within the VDP file or in one or more separate files. The RIP software associates the variable data with the appearance information defined for the variable data area, as described further in element (e) below.
(e) applying said graphics state corresponding to said variable data area to said variable data to generate a variable data bit map; and	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the

'665 Patent, Claim 1	

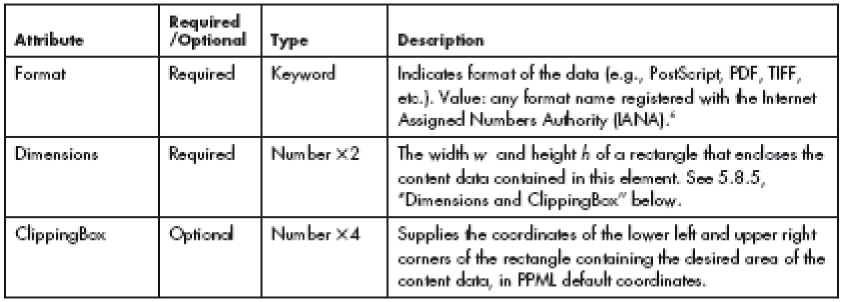
order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34.

"The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.

"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.

"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.

"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).⁴
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text

'665 Patent, Claim 1	
	x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) merging said variable data bit map with said bit map of said template;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of

'665 Patent, Claim 1	
	content for a given printed page. Ref. [15] at 4-5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
wherein said graphics state corresponding to said variable data area is applied repeatedly to variable data to generate a multitude of variable data bit maps without the need to repeat said executing step (b).	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15.

'665 Patent, Claim 1	
	PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

'665 Patent, Claim 12	
12. The method of claim 1 wherein said reserving, retrieving, associated, applying, and merging steps are repeated for each variable data area defined by said page description code.	VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving,

'665 Patent, Claim 12	
	retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'665 Patent, Claim 13	
13. The method of claim 12 wherein said reserving, retrieving, associating, applying and merging steps are activated by a control task running in a printer controller, and wherein said control task interrupts said page description code execution upon identifying a predetermined command in said page description code.	As described above, the steps of reserving, retrieving, associating, applying and merging are all activated and monitored by a control task running in a dedicated print server or digital front end associated with the press. Each of the respective print servers or digital front ends runs RIP software such as the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The control task interrupts said page description code execution upon identifying a predetermined command in said page description code to enable other operations to be performed.

'665 Patent, Claim 20	

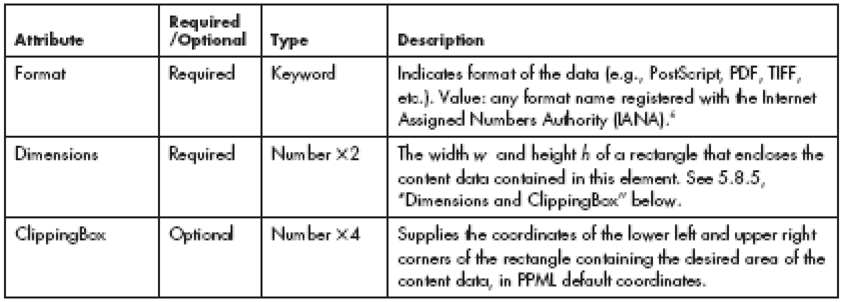
‘665 Patent, Claim 20	
20. A method for generating a plurality of bit maps suitable for high-speed printing or plate-making from a page description code representing a template and defining at least one variable data area, and from a merge file containing a plurality of data records of at least one variable data field type, the method comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor (“RIP”) software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the “executing a control task” step below. Each of these files further defines at least one variable data area, as described further in the “executing a control task” step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP’s digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. To the extent that third-parties, such as Cenveo’s customers and/or their print media agents, perform the step of generating these files, Cenveo directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo enters contracts with these third parties through which Cenveo enforces the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data.

‘665 Patent, Claim 20	
	For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and

‘665 Patent, Claim 20

orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Cenveo runs software on dedicated print servers or digital front ends, as described above, to

'665 Patent, Claim 20	
	retrieve variable data elements stored in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be referenced from outside of the PPMLT file itself. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a page description code interpretive program, said interpretive program generates graphics states for each data area defined by said page description code;	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar

'665 Patent, Claim 20	
	software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The RIP software necessarily includes a module or other discrete software component that interprets VDP files, including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data.

'665 Patent, Claim 20

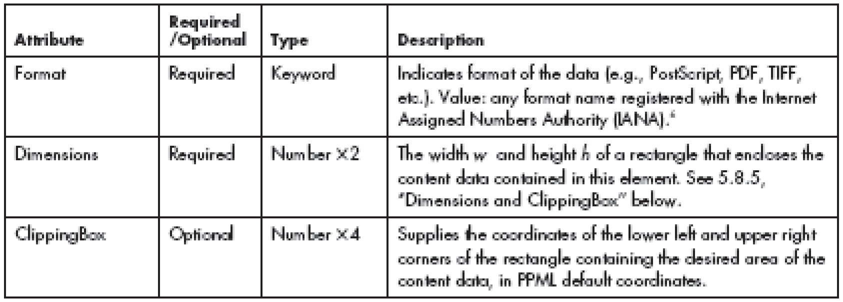
The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34.

"The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.

"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.

"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.

"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing,

‘665 Patent, Claim 20	
	size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a control task in conjunction with said interpretive program, said control task identifies said variable data area defined by said page description code and reserves said graphics states generated by said interpretive program for said variable data area, said control task generates a template bit map defined by said page description code, and after the completion of said interpretive program, said control task saves said template bit map in memory; and	As described above, Cenveo runs software on dedicated print servers or digital front ends. RIP software identifies variable data areas defined in VDP files. The RIP software necessarily includes one or more module or other discrete software component that identifies variable data areas, reserves graphics states, and generates one or more template bitmap, and saves one or more template bitmap in memory. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only

'665 Patent, Claim 20	
	rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable

'665 Patent, Claim 20	
	data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element.

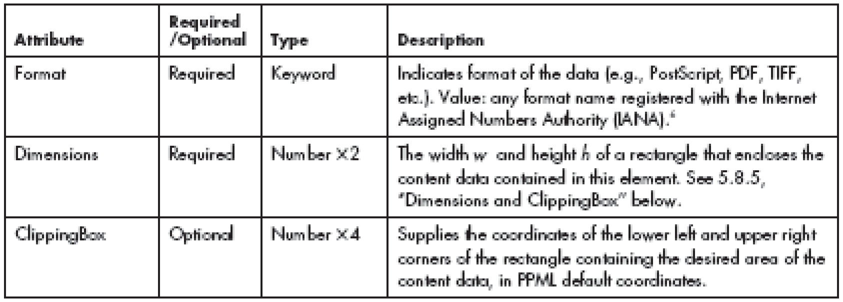
‘665 Patent, Claim 20	
	Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a merge task upon completion of said interpretive program, said merge task generates variable data bit maps for said data records in said merge file by applying said reserved graphics states to said data records, and said merge task merges said variable data bit maps with a separate copy of said template bit map to create the plurality of bit maps suitable for high-speed printing or plate making;	As described above, Cenveo runs software on dedicated print servers or digital front ends. RIP software applies appearance information found in the VDP file to the corresponding variable data areas. The RIP software necessarily includes one or more module or other discrete software component that applies the appearance information to the corresponding variable data to generate a variable data bit map. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

‘665 Patent, Claim 20	

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number × 2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number × 4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate

'665 Patent, Claim 20	
	image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Cenveo runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
whereby said reserved graphics states are applied repeatedly to said data records to generate said variable data bit maps for said data records without the need to repeat said steps of	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in

'665 Patent, Claim 20	
executing a page description code interpretive program and executing a control task in conjunction with said interpretive program.	the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

U.S. Patent No. 5,937,153 ("the '153 patent")

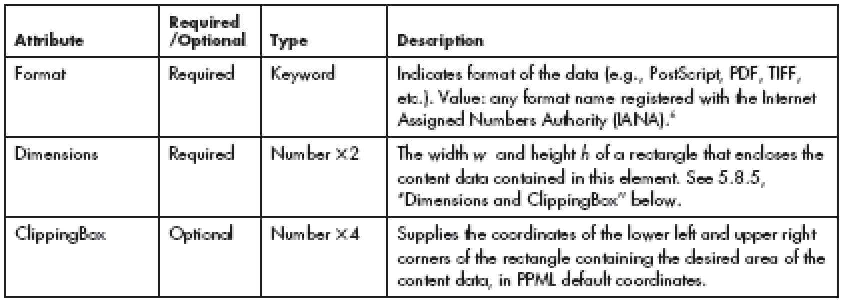
'153 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code specification, the page description code specification defining at least one data area to become variable, and the page description code further defining a graphics state corresponding to the data area, the graphics state including at least one attribute which controls the appearance of data in the data area;	Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in element (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartSTream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. To the extent that third-parties, such as Cenveo's customers and/or their print media agents, perform the step of generating these files, Cenveo directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo enters contracts with these third parties through which Cenveo enforces the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data.

'153 Patent, Claim 1	
	For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and

'153 Patent, Claim 1	

orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.

"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

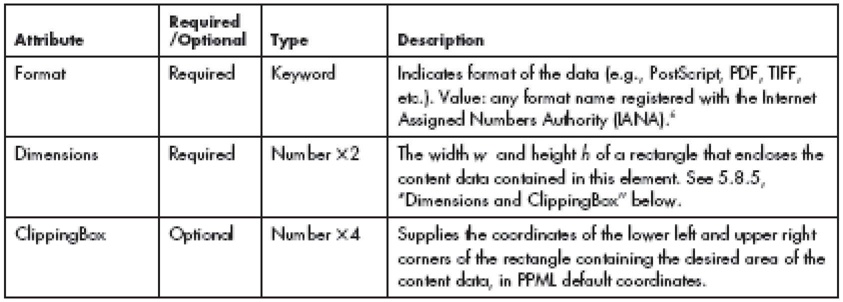
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Cenveo may use other VDP file types with infringing characteristics, features, and functions

'153 Patent, Claim 1	
	similar to those described above in these exemplary file types.
(b) interpreting the page description code specification, and during the interpretation, identifying the data area defined by the page description code specification;	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to

'153 Patent, Claim 1	
	"content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) storing the graphics state corresponding to the data area upon the identification of the variable data area in step (b);	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) retrieving a variable data item from a plurality of variable data items;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable

'153 Patent, Claim 1	
	data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) applying the stored graphics state to the variable data item to generate a variable data bit map; and	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data.

'153 Patent, Claim 1	
	The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.  Ref. [11] at 28; Ref. [12] at 40.

Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

‘153 Patent, Claim 1	
	In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) repeating steps (d) and (e) for remaining variable data items in the plurality of variable data items, whereby the stored graphics state is applied repeatedly to generate a plurality of variable data bit maps.	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those

'153 Patent, Claim 1	
	objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 3	
3. The computer implemented method of claim 1, wherein the page description code specification represents a template and includes a static	As described for claim 1 of the '153 patent, Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files. Each of these files represents a template. These VDP files use static data areas to quickly manage VDP jobs. PPML for example, performs more efficiently when the static data areas are defined in advance. Ref. [10] at 10.

'153 Patent, Claim 3	
data area; and the computer implemented method further comprises the steps of:	
executing portions of the page description code specification corresponding to the static data area to generate a template bit map;	Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the

'153 Patent, Claim 3	
	Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
storing the template bit map; and	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped

'153 Patent, Claim 3	
	(prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
merging each of the plurality of the variable data bit maps into a clean copy of the template bit map to create a plurality of merged bit maps.	Cenveo runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment.

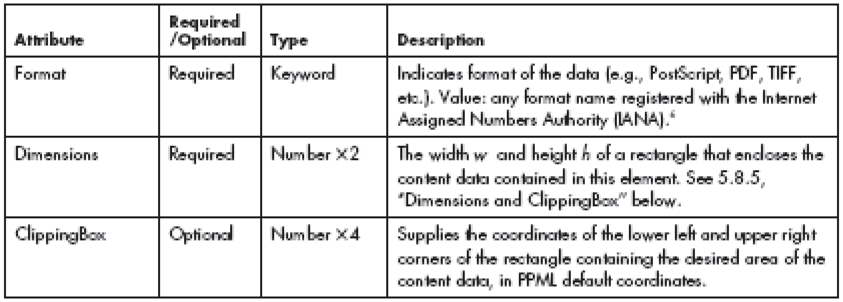
'153 Patent, Claim 3	
	The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 4	
4. The computer implemented method of claim 1, wherein the identifying step includes the step of detecting predefined characters within a text string defined in the page description code specification.	As described for claim 1 of the '153 patent, the controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files use objects denoted by the text "obj" to identify template and variable data areas. Further, the text "/XObject" denotes information in certain objects that will be reused. The RIP software may detect these characters or the RIP software may evaluate repetitive text within the PDF files to identify data areas. In PDF/VT, XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." [17] at § 6.7.3. For example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point

'153 Patent, Claim 4	
	to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. Each of these structures is associated with a predetermined characters within the JLYT file. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 5	
5. The computer implemented method of claim 1, wherein the attribute is a size attribute, a font attribute, a position attribute, an orientation attribute or a location attribute.	As described above, PDF, PDF/VT, PPML, PPMLT, and JLYT can each define appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The

‘153 Patent, Claim 5	
	format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39. “The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'153 Patent, Claim 5</th><th></th></tr>
<tr><td></td><td><table><tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr><tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr><tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr><tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr></table>
Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</td></tr>
</table>

<table>
<tr><th>'153 Patent, Claim 6</th><th></th></tr>
<tr><td>6. A computer implemented method for processing a page description code specification comprising the steps of:</td><td>Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating,</td></tr>
</table>

'153 Patent, Claim 6	
	referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further below. Each of these files further defines at least one variable data area, as described further in the "interpreting" step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5.

'153 Patent, Claim 6	
	In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.
interpreting the page description code specification, and during the interpretation, identifying a data area defined by the page description code specification;	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one

'153 Patent, Claim 6	
	or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
upon the identification of the data area, storing a graphics	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing,

'153 Patent, Claim 6	
state set forth in the page description code specification which defines an attribute of how data is to appear in the data area; and	rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
repeatedly retrieving data records from a plurality of data records and applying the stored graphics state to the data records to generate a plurality of bitmaps of the data records so that the bitmaps of the data records include the attribute.	Cenveo runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or

'153 Patent, Claim 6	
	externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the

‘153 Patent, Claim 6	

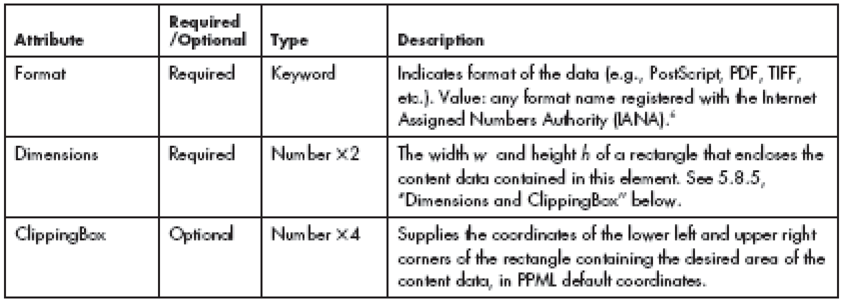
order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[a]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text

'153 Patent, Claim 6	
	x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page.

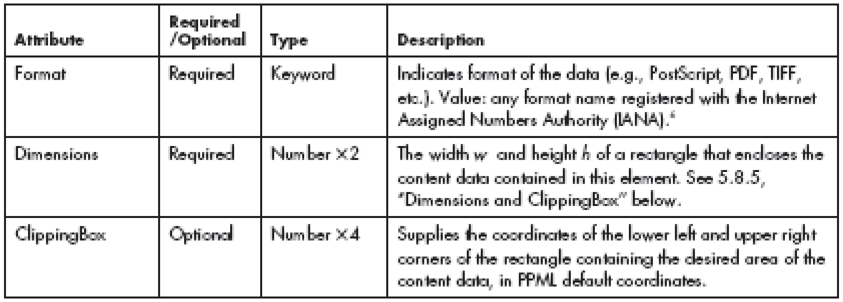
'153 Patent, Claim 6	
	PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

U.S. Patent No. 6,381,028 ("the '028 patent")

'028 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description code specification, the page description code specification defining at least one data area, and the page description code further defining a graphics state including at least one attribute which controls the appearance of data in the data area;	Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in step (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and

‘028 Patent, Claim 1	
	skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39. “The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and

<table>
<tr><th>'028 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>
are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(b) interpreting the page description code specification, and during the interpretation step, identifying the data area</td><td>Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP</td></tr>
</table>

'028 Patent, Claim 1	
defined by the page description code specification;	SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions

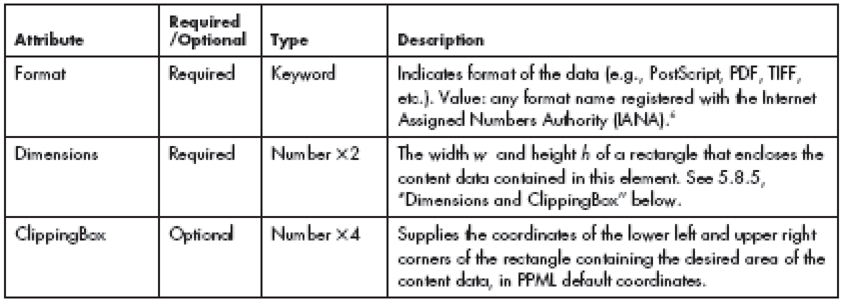
'028 Patent, Claim 1	
	similar to those described above in these exemplary file types.
(c) upon the identification of the data area in step (b), applying the graphics state corresponding to the data area to a set of alphanumeric characters so as to generate a plurality of character bit maps;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to characters associated with the variable data areas. The appearance information is applied to the characters by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.

‘028 Patent, Claim 1	

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate

'028 Patent, Claim 1	
	image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. RIP software applies the graphics state as part of generating character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) storing the plurality of character bit maps;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to store character bitmaps. The character bitmaps are stored by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software stores the character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are

'028 Patent, Claim 1	
	likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) retrieving a variable data item from a plurality of variable data items;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) associating the variable data item with the plurality of	Cenveo runs software on dedicated print servers or digital front ends, as described above, to associate variable data items with the character bitmaps. The variable data items associated with

'028 Patent, Claim 1	
character bit maps;	character bitmaps are identified by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software necessarily associates the character bitmaps for each character in the respective variable data areas. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a variable data bit map for the variable data using the character bit maps; and	Cenveo runs software on dedicated print servers or digital front ends, as described above, to generate variable data bitmaps. The variable data bitmaps are generated by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that

'028 Patent, Claim 1	
	reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(h) repeating steps (e) through (g) for remaining variable data items in the plurality of variable data items, whereby the stored character bit maps are used repeatedly to generate a plurality of variable data bit maps.	Cenveo runs software on dedicated print servers or digital front ends, as described above, to use the stored character bitmaps for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the stored character bitmaps and the template bitmap are reused for each instance of the document. As discussed above, RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'028 Patent, Claim 2	
2. The computer implemented method of claim 1, wherein the page description code specification represents a	The elements of claim 1 are described in the chart above. Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a

'028 Patent, Claim 2	
template and includes a static data area, and the computer implemented method further comprises the steps of:	template and includes a static data area, as described further in the "executing" step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9].
executing portions of the page description code specification corresponding to the static data area to generate a template bit map; and	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can

'028 Patent, Claim 2	
	be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.
merging each of the plurality of the variable data bit maps into clean copies of the template bit map to create a plurality of merged bit maps.	Cenveo runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment.

'028 Patent, Claim 2	
	The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5.

'028 Patent, Claim 4	
4. A computer implemented method for generating a reusable template bit map suitable for high-speed variable printing, comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the "executing" step below.
generating a page description code specification, the page description code specification	Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a

'028 Patent, Claim 4	
defining at least one variable data area and at least one static data area;	template, as described further in the "executing" step below. Each of these files further defines at least one variable data area, as described further in the "identifying" step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartSTream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. To the extent that third-parties, such as Cenveo's customers and/or their print media agents, perform the step of generating these files, Cenveo directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo enters contracts with these third parties through which Cenveo enforces the obligations that it imposes upon third-parties. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description code specification, and during the interpreting step,	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers.
generating a bitmap of the static data area and adding the bitmap of the static data area to a template bitmap;	Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5.

'028 Patent, Claim 4	
	For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's

'028 Patent, Claim 4	
	appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
identifying the variable data area, and	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata

'028 Patent, Claim 4	
	that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
responsive to the identification of the variable data, not adding a bitmap of the variable data area to the template bitmap; and	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets, and therefore does not include a bitmap of the variable data area. Adding a bitmap of the variable data area to the template bitmap would prevent reuse of the static bitmap. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
saving the template bitmap, whereby copies of the template bitmap can be continuously accessed to create a plurality of variable data bitmaps.	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used

'028 Patent, Claim 4	
	repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

U.S. Patent No. 7,274,479 ("the '479 patent")

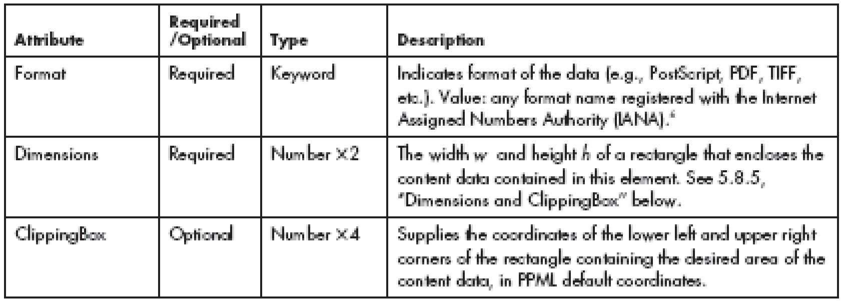
'479 Patent, Claim 9	
9. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing, comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a print specification, the print specification defining at least one variable data area and at least one static data area;	Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a static data area, as described further below. Each of these files further defines at least one variable data area, as described further in element (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5.

'479 Patent, Claim 9	
	For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) providing a plurality of	Cenveo runs software on dedicated print servers or digital front ends, as described above, to

'479 Patent, Claim 9	
variable data items;	retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) identifying the variable data area;	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar

'479 Patent, Claim 9	
	software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Cenveo uses such print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data, as described above in element (a). The VDP file defines variable data areas based on the surrounding tags of the data element.
(d) associating a graphic state with the variable data area, the graphic state including at least one attribute controlling the appearance of items to be printed in the variable data area;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to associate appearance information found in the VDP file to the corresponding variable data. The VDP file includes information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be

'479 Patent, Claim 9	
	printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>
<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

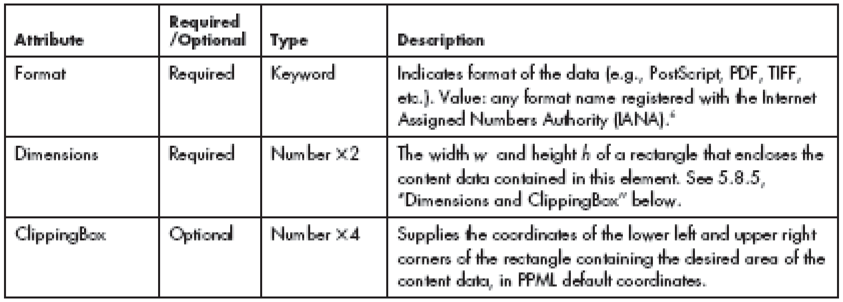
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

As described above in element (b), variable data may be stored within the VDP file or in one or more separate files. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field.

Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</td></tr>
<tr><td>(e) retrieving a variable data item from the plurality of</td><td>Cenveo runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The</td></tr>
</table>

'479 Patent, Claim 9	
variable data items;	variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) generating a bitmap for the variable item, the generating step including a step of applying the graphic state associated with the variable data area to the variable data item; and	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The

'479 Patent, Claim 9	
	graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p>
</td></tr>
<tr><td>(g) repeating steps (e) and (f) for remaining variable data items in the plurality of variable data items, whereby the graphic state associated with the variable data area is applied</td><td>
<p>Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document.</p>
<p>The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data</p>
</td></tr>
</table>

'479 Patent, Claim 9	
repeatedly to generate a plurality of variable data bitmaps.	bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions

'479 Patent, Claim 9	
	similar to those described above in these exemplary file types.

'479 Patent, Claim 10	
10. The method of claim 9, wherein the graphic state associated with the variable data area is defined within the print specification.	Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data, as discussed above with respect to element (d) of claim 9 of the '479 Patent. The appearance information may be defined within the print specification either by referencing an external file or by providing the appearance information directly within the VDP file.

'479 Patent, Claim 15	
15. The method of claim 9, wherein the variable data area and the static data area are defined, at least in part, by page description language commands.	As described for claim 9 of the '479 Patent, the VDP file defines static and variable data areas based on the surrounding tags of the data element. VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects each incorporate page description language commands. Each of these files is a page description language file, and the tags and commands included in each of these files are therefore page description language commands. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing, as described in elements (a) and (b) of claim 9. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 17	
17. The method of claim 9,	A static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets.

'479 Patent, Claim 17	
further comprising a step of caching a representation of the static data area, whereby the cached representation of the static data area is available for merging with the variable data bitmaps to generate merged documents.	By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server

'479 Patent, Claim 17	
	or digital front end. *See* Ref. [15] at 5.

'479 Patent, Claim 18	
18. The method of claim 17, wherein the cached representation of the static data area is a bitmap representation.	The cached representation of the static data area is a bitmap to avoid the redundant burden of the system to continually compute the contents appearance, as discussed above for claim 17 of the '479 Patent. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it

'479 Patent, Claim 18	
	takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 19	
19. The method of claim 9, wherein: the plurality of data items are associated with a field name; and	As described for claim 9 of the '479 Patent, each field retrieved from a variable data record is matched to a corresponding named variable data area defined within the VDP file.
the step of identifying a variable data area includes the step of detecting, in the print specification, a character string associated with the variable data area that matches the field name associated with the plurality of data items.	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, in PDF/VT files, document part metadata provides field name information for variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data. Ref. [12] at 42-43. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition,

'479 Patent, Claim 19	
	PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.

U.S. Patent No. 7,333,233 ("the '233 patent")

'233 Patent, Claim 12	
12. A computer implemented method for generating a static bitmap suitable for high-speed variable printing, comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below, and in the "interpreting" step. Each of these files further defines at least one variable data area, as described further below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas.

'233 Patent, Claim 12	
	In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3.

'233 Patent, Claim 12	
	In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description language file, and during the interpreting step, generating a static bitmap of the static data area;	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar

'233 Patent, Claim 12	
	software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. The VDP file defines static data areas based on the surrounding tags of the data element. The type

'233 Patent, Claim 12	
	of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
and saving the static bitmap, whereby the saved static bitmap is used repeatedly in the generation of a plurality of documents, each of which contains the static bitmap and a variable data bitmap.	Cenveo runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12]

'233 Patent, Claim 12	
	at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and

'233 Patent, Claim 12	
	can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving, retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional

'233 Patent, Claim 12	
	variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'233 Patent, Claim 14	
14. A computer implemented method for generating a plurality of bitmaps suitable for high-speed printing, comprising the steps of:	Defendant Cenveo, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Cenveo operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Cenveo operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Cenveo operates digital presses manufactured by HP, including: Indigo w3050, Indigo w3250, Indigo 5000, and Indigo 7200. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	Cenveo operates software tools as part of a process by which Cenveo generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below. Each of these files further defines at least one variable data area, as described further below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated

'233 Patent, Claim 14	
	at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Cenveo. Refs. [3]-[9]. The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant

'233 Patent, Claim 14	
	downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5.

'233 Patent, Claim 14	
	JLYT files include channels that define links to variable content. Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) providing a merge file including a plurality of variable data items;	The VDP files can use variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which stores data either internally or externally. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. Ref. [17] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) processing the page description language file, and during the processing step, generating a static bitmap of the static data area and associating the variable data area with the plurality of variable data items; and	Cenveo runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Cenveo includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Cenveo uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap

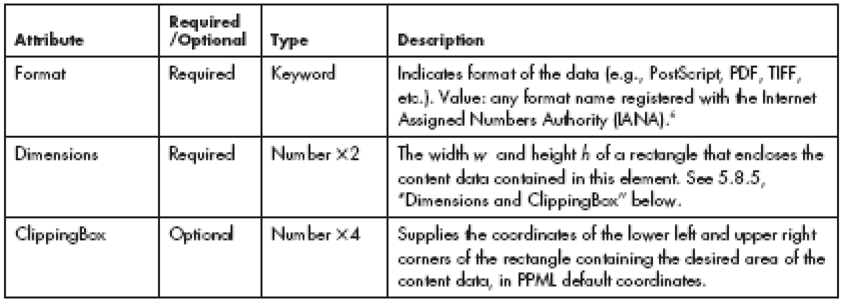
'233 Patent, Claim 14	
	comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.

'233 Patent, Claim 14	
	The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Cenveo runs software on dedicated print servers or digital front ends, as described above, to associate variable data areas with variable data items. For example, in PDF/VT files, document part metadata provides field name information for variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data.

'233 Patent, Claim 14	
	Ref. [12] at 42-43. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.
(d) saving the static bitmap;	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref.

'233 Patent, Claim 14	
	[11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) generating a first variable data bitmap of a first one of the variable data items utilizing a graphics state associated with the variable data area;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format,

'233 Patent, Claim 14	
	dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'233 Patent, Claim 14</th><th></th></tr>
<tr><td></td><td>
<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width <i>w</i> and height <i>h</i> of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</td></tr>
<tr><td>(f) merging the first variable data bitmap with a copy of the static bitmap to produce a first output bitmap;</td><td>Cenveo runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. <i>See</i> Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap.

For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient.</td></tr>
</table>

'233 Patent, Claim 14	
	Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. Cenveo may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a next variable data bitmap of a next one of the variable data items utilizing a graphics state associated with the variable data area;	Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document, according to the contentions with respect to element (e). Appearance information is reused for each instance of the document. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. Cenveo runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data

'233 Patent, Claim 14	
	bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Cenveo may use other VDP file types with infringing characteristics, features, and functions

'233 Patent, Claim 14	
	similar to those described above in these exemplary file types.
and (h) merging the next variable data bitmap with a copy of the static bitmap to produce a next output bitmap;	The print server or digital front end merges the variable data bitmaps with the template bitmap according to the contentions with respect to element (f). The appearance information and the template bitmap are reused for each instance of the document. The template bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. The template bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins.

'233 Patent, Claim 14	
	PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2.
and (i) repeating steps (g) (h) for remaining variable data items in the plurality of variable data items.	The activities performed for steps (g) and (h) are repeated for each remaining variable data item in the plurality of data items.

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. These contentions are based solely on publicly available information relating to exemplary variable data ("VDP") files and raster image processor (RIP") software and press control software used on HP's presses. Discovery in this case has not yet begun, and the charts below do not reflect any information produced by defendants Cenveo, Inc. and Hewlett-Packard Company. Other VDP files, RIP software and press control software may be identified through discovery as being used by defendants. IPT reserves the right to support its contentions with additional material produced by the defendants or subsequently identified by IPT.

[1] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf
[2] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf
[3] HP T200 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T200_Data_Sheet.pdf
[4] HP T300 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T300_Data_Sheet.pdf
[5] HP T350 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T350_Data_Sheet.pdf
[6] HP T400 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T400_Data_Sheet.pdf
[7] HP Indigo w3250 Data Sheet, available at http://h10010.www1.hp.com/wwpc/pscmisc/vac/us/product_pdfs/90566.pdf
[8] HP Indigo 5600 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2012/HPPredrupa12/HP_Indigo_5600.pdf
[9] HP Indigo 7500 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2010/IPEX2010/HP_Indigo_7500_DS.PDF
[10] PPML Template, available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html
[11] PPML Specification v1.5, available at http://www.standards.podi.org/ppml/specification.html
[12] PPML Specification v2.1, available at http://www.standards.podi.org/ppml/specification.html

[13] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf

[14] HP Indigo Yours Truly Designer 7 User Guide

[15] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.

[16] Adobe Systems Inc., PDF Reference 5th Ed., v. 1.6, available at http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference16.pdf

[17] ISO 16612-2:2010, available at http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=46428

[18] Global Graphics, Do PDF/VT Right, available at http://www.globalgraphics.com/doPDFVTright/

U.S. Patent No. 5,729,665 ("the '665 patent")

'665 Patent, Claim 1	
1. A method for generating multiple bit maps suitable for high-speed printing or plate-making comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '665 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the '665 patent and of the technology claimed within the '665 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '665 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage

‘665 Patent, Claim 1	
	direct infringement of the ‘665 patent. HP, Cenveo, and HP’s other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services and resulting printed products to their customers. HP, Cenveo, and HP’s other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP’s other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor (“RIP”) software provided by HP or a third-party. For example, HP, Cenveo, and HP’s other customers operate digital presses manufactured by HP, including without limitation HP’s Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code representing a template, said page description code defining at least one variable data area and said page description code further defining a graphics state corresponding to said variable data area, said graphics state including at least one attribute which controls the appearance of variable data in said variable data area;	HP and Cenveo and other HP customers operate software tools as part of a process by which HP, Cenveo, and HP’s other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in element (b) below. Each of these files further defines at least one variable data area, as described further in element (b) below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP’s digital presses such as the ones operated by HP and Cenveo and other HP customers. Refs. [3]-[9].

'665 Patent, Claim 1	
	To the extent that third-parties, such as Cenveo's customers and/or their print media agents, perform the step of generating these files, Cenveo and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo and HP's other customers enter contracts with these third parties through which Cenveo and HP's other customers enforce the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed

‘665 Patent, Claim 1

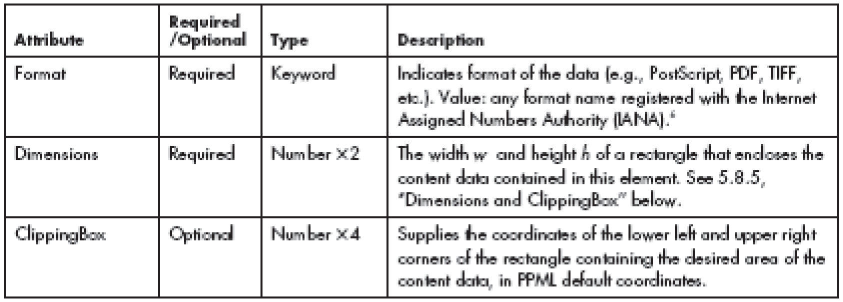
on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-

'665 Patent, Claim 1	
	spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) executing said page description code to generate a bit map of said template, and during said execution, identifying said variable data area defined by said page description code and reserving said graphics state corresponding to said variable data area upon said identification;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the

‘665 Patent, Claim 1	
	document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and

'665 Patent, Claim 1	
	treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT

'665 Patent, Claim 1	
	include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) retrieving variable data;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

'665 Patent, Claim 1	
	types.
(d) associating said variable data with said graphics state corresponding to said variable data area;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to associate the appearance information found in the VDP file to the corresponding variable data. As described above, variable data may be stored within the VDP file or in one or more separate files. The RIP software associates the variable data with the appearance information defined for the variable data area, as described further in element (e) below.
(e) applying said graphics state corresponding to said variable data area to said variable data to generate a variable data bit map; and	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the

‘665 Patent, Claim 1	

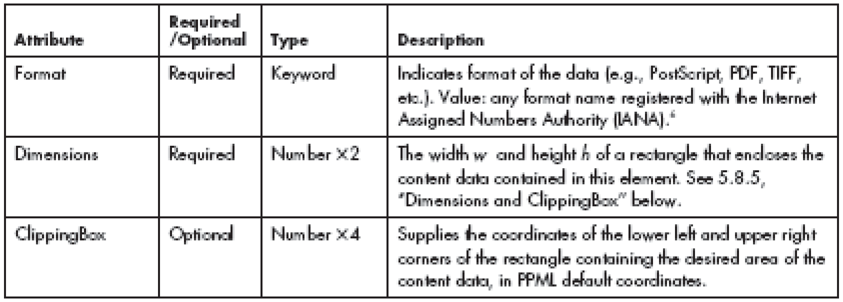
order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text

'665 Patent, Claim 1	
	x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) merging said variable data bit map with said bit map of said template;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of

‘665 Patent, Claim 1	
	content for a given printed page. Ref. [15] at 4-5. HP, Cenveo, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
wherein said graphics state corresponding to said variable data area is applied repeatedly to variable data to generate a multitude of variable data bit maps without the need to repeat said executing step (b).	HP, Cenveo, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more

‘665 Patent, Claim 1	
	variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

‘665 Patent, Claim 12	
12. The method of claim 1 wherein said reserving, retrieving, associated, applying, and merging steps are repeated for each variable data area defined by said page description code.	VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page.

'665 Patent, Claim 12	
	PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving, retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'665 Patent, Claim 13	
13. The method of claim 12 wherein said reserving, retrieving, associating, applying and merging steps are activated by a control task running in a printer controller, and wherein said control task interrupts said page description code execution upon identifying a predetermined command in said	As described above, the steps of reserving, retrieving, associating, applying and merging are all activated and monitored by a control task running in a dedicated print server or digital front end associated with the press. Each of the respective print servers or digital front ends runs RIP software such as the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The control task interrupts said page description code execution upon identifying a predetermined command in said page description code to enable other operations to be performed.

‘665 Patent, Claim 13	
page description code.	

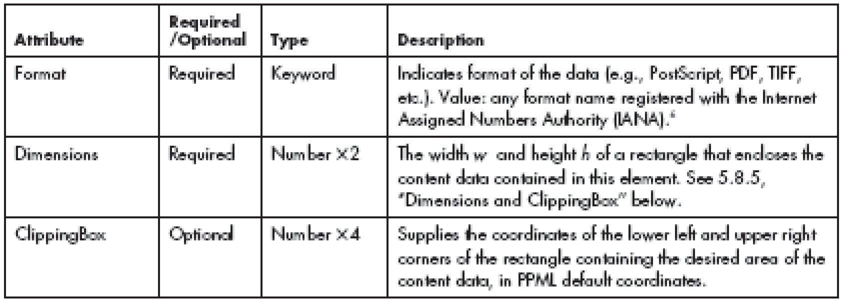
‘665 Patent, Claim 20	
20. A method for generating a plurality of bit maps suitable for high-speed printing or plate-making from a page description code representing a template and defining at least one variable data area, and from a merge file containing a plurality of data records of at least one variable data field type, the method comprising the steps of:	Defendant Hewlett-Packard (“HP”), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print (“VDP”) jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo’s facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP’s Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the ‘665 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the ‘665 patent and of the technology claimed within the ‘665 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the ‘665 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the ‘665 patent. HP, Cenveo, and HP’s other customers directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services and resulting printed products to their customers. HP, Cenveo, and HP’s other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP’s other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor (“RIP”) software provided by HP or a third-party. For example, HP,

'665 Patent, Claim 20	
	Cenveo, and HP's other customers operate digital presses manufactured by HP, , including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the "executing a control task" step below. Each of these files further defines at least one variable data area, as described further in the "executing a control task" step below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. To the extent that third-parties, such as Cenveo's customers and/or their print media agents, perform the step of generating these files, Cenveo and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo and HP's other customers enter contracts with these third parties through which Cenveo and HP's other customers enforce the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or

'665 Patent, Claim 20	
	PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.

‘665 Patent, Claim 20

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP

'665 Patent, Claim 20	
	or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be referenced from outside of the PPMLT file itself. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a page description code interpretive program, said interpretive program generates graphics states for each data area defined by said page description code;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo,

'665 Patent, Claim 20	
	or Esko installed on HP's print servers or digital front end computers. The RIP software necessarily includes a module or other discrete software component that interprets VDP files, including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data.

‘665 Patent, Claim 20	

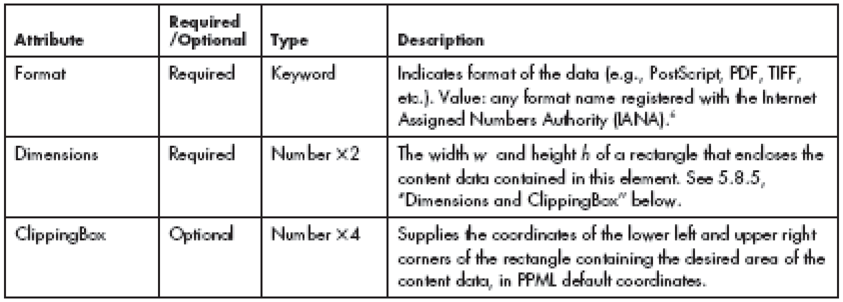
The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing,

'665 Patent, Claim 20	
	size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a control task in conjunction with said interpretive program, said control task identifies said variable data area defined by said page description code and reserves said graphics states generated by said interpretive program for said variable data area, said control task generates a template bit map defined by said page description code, and after the completion of said interpretive program, said control task saves said template bit map in memory; and	As described above, HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends. RIP software identifies variable data areas defined in VDP files. The RIP software necessarily includes one or more module or other discrete software component that identifies variable data areas, reserves graphics states, and generates one or more template bitmap, and saves one or more template bitmap in memory. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet

‘665 Patent, Claim 20	
	presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata

'665 Patent, Claim 20	
	that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT

'665 Patent, Claim 20	
	files define stored channels that include scaling and rotation parameters for each element. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a merge task upon completion of said interpretive program, said merge task generates variable data bit maps for said data records in said merge file by applying said reserved graphics states to said data records, and said merge task merges said variable data bit maps with a separate copy of said template bit map to create the plurality of bit maps suitable for high-speed printing or plate making;	As described above, HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends. RIP software applies appearance information found in the VDP file to the corresponding variable data areas. The RIP software necessarily includes one or more module or other discrete software component that applies the appearance information to the corresponding variable data to generate a variable data bit map. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how

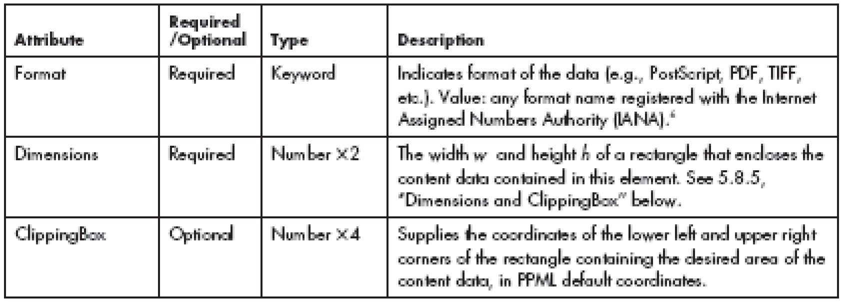
‘665 Patent, Claim 20

a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is

'665 Patent, Claim 20	
	the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
whereby said reserved graphics states are applied repeatedly to	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the

'665 Patent, Claim 20	
said data records to generate said variable data bit maps for said data records without the need to repeat said steps of executing a page description code interpretive program and executing a control task in conjunction with said interpretive program.	variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54.

‘665 Patent, Claim 20	
	In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

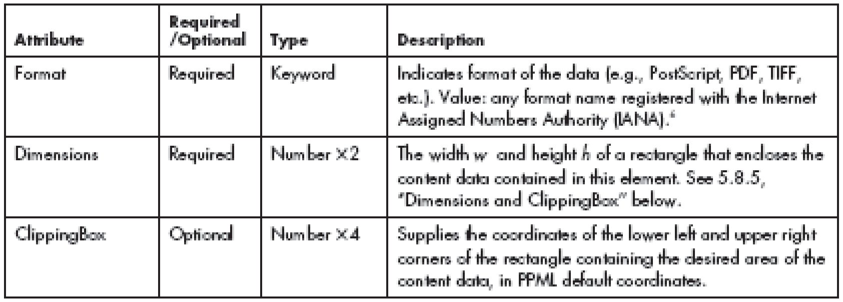
U.S. Patent No. 5,937,153 ("the '153 patent")

'153 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '153 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the '153 patent and of the technology claimed within the '153 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '153 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '153 patent. HP, Cenveo, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Cenveo, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Cenveo, and HP's other customers operate digital presses manufactured by HP, including without

'153 Patent, Claim 1	
	limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code specification, the page description code specification defining at least one data area to become variable, and the page description code further defining a graphics state corresponding to the data area, the graphics state including at least one attribute which controls the appearance of data in the data area;	HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in element (b) below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. To the extent that third-parties, such as Cenveo's customers and/or their print media agents, perform the step of generating these files, Cenveo and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo and HP's other customers enter contracts with these third parties through which Cenveo and HP's other customers enforce the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and

'153 Patent, Claim 1	
	skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and

<table>
<tr><th>'153 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>
are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

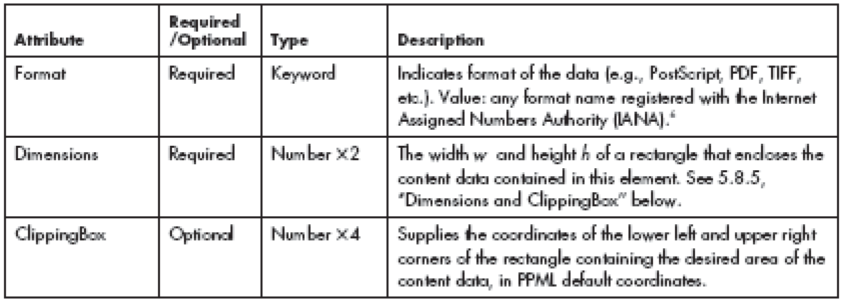
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(b) interpreting the page description code specification, and during the interpretation,</td><td>HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of</td></tr>
</table>

'153 Patent, Claim 1	
identifying the data area defined by the page description code specification;	executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5.

'153 Patent, Claim 1	
	HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) storing the graphics state corresponding to the data area upon the identification of the variable data area in step (b);	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) retrieving a variable data item from a plurality of variable data items;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or

‘153 Patent, Claim 1	
	externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., “<xsl: value-of select=’name’/>” points to a database entry for the “name” element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Cenveo, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) applying the stored graphics state to the variable data item to generate a variable data bit map; and	HP, Cenveo, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP’s print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data.

<table>
<tr><th>'153 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>
<p>The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34.</p>
<p>"The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.</p>
<p>"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.</p>
<p>"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.</p>
<p>"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.</p>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width <i>w</i> and height <i>h</i> of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
</td></tr>
</table>

‘153 Patent, Claim 1	
	In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Cenveo, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) repeating steps (d) and (e) for remaining variable data items in the plurality of variable data items, whereby the stored graphics state is applied repeatedly to generate a plurality of variable data bit maps.	HP, Cenveo, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each

'153 Patent, Claim 1	
	recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 3	
3. The computer implemented method of claim 1, wherein the	As described for claim 1 of the '153 patent, HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files.

‘153 Patent, Claim 3	
page description code specification represents a template and includes a static data area; and the computer implemented method further comprises the steps of:	Each of these files represents a template. These VDP files use static data areas to quickly manage VDP jobs. PPML for example, performs more efficiently when the static data areas are defined in advance. Ref. [10] at 10.
executing portions of the page description code specification corresponding to the static data area to generate a template bit map;	HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are

‘153 Patent, Claim 3	
	explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. HP, Cenveo, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
storing the template bit map; and	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. “Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is

'153 Patent, Claim 3	
	central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
merging each of the plurality of the variable data bit maps into a clean copy of the template bit map to create a plurality of merged bit maps.	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5.

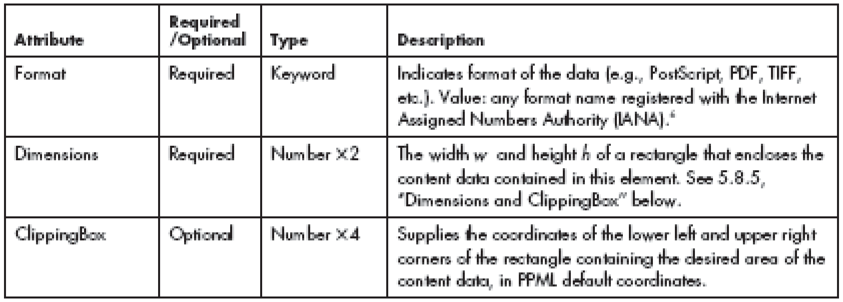
‘153 Patent, Claim 3	
	In another example, “PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction.” Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define “channels” that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. HP, Cenveo, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

‘153 Patent, Claim 4	
4. The computer implemented method of claim 1, wherein the identifying step includes the step of detecting predefined characters within a text string defined in the page description code specification.	As described for claim 1 of the ‘153 patent, the controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files use objects denoted by the text “obj” to identify template and variable data areas. Further, the text “/XObject” denotes information in certain objects that will be reused. The RIP software may detect these characters or the RIP software may evaluate repetitive text within the PDF files to identify data areas. In PDF/VT, XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” [17] at § 6.7.3. For example, within a PPML file the OBJECT tag “associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a

'153 Patent, Claim 4	
	REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. Each of these structures is associated with a predetermined characters within the JLYT file. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 5	
5. The computer implemented method of claim 1, wherein the attribute is a size attribute, a font attribute, a position attribute, an orientation attribute or a location attribute.	As described above, PDF, PDF/VT, PPML, PPMLT, and JLYT can each define appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character

'153 Patent, Claim 5	
	spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'153 Patent, Claim 5</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p>
</td></tr>
</table>

'153 Patent, Claim 6	
6. A computer implemented method for processing a page description code specification	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers,

'153 Patent, Claim 6	
comprising the steps of:	product demonstrations, open houses and at Cenveo's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '153 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the '153 patent and of the technology claimed within the '153 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '153 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '153 patent. HP, Cenveo, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Cenveo, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Cenveo, and HP's other customers operate digital presses manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.

'153 Patent, Claim 6	
	HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further below. Each of these files further defines at least one variable data area, as described further in the "interpreting" step below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are

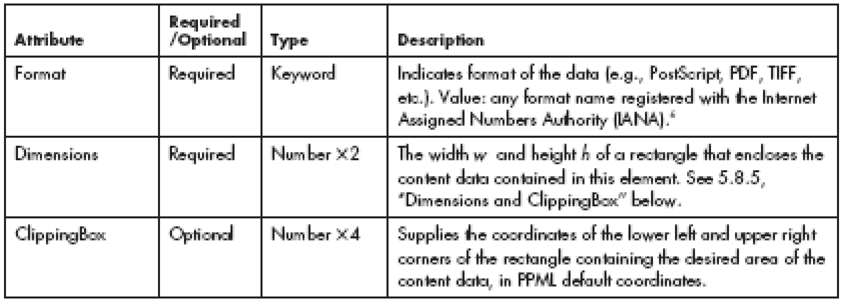
'153 Patent, Claim 6	
	explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.
interpreting the page description code specification, and during the interpretation, identifying a data area defined by the page description code specification;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects.

‘153 Patent, Claim 6	
	The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag “associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to “content packages” that “include any static content in the file (text and image page objects, for instance).” Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
upon the identification of the data area, storing a graphics state set forth in the page description code specification	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some

'153 Patent, Claim 6	
which defines an attribute of how data is to appear in the data area; and	or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
repeatedly retrieving data records from a plurality of data records and applying the stored graphics state to the data records to generate a plurality of bitmaps of the data records so that the bitmaps of the data records include the attribute.	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or

'153 Patent, Claim 6	
	externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the

‘153 Patent, Claim 6	
	order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39. “The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text

'153 Patent, Claim 6	
	x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics

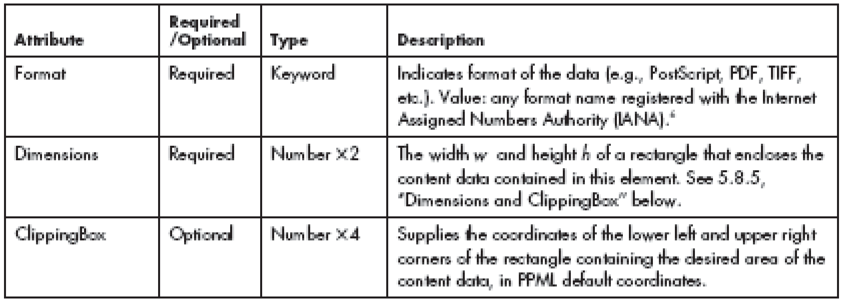
'153 Patent, Claim 6	
	state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

U.S. Patent No. 6,381,028 ("the '028 patent")

'028 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '028 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the '028 patent and of the technology claimed within the '028 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '028 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '028 patent. HP, Cenveo, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Cenveo, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Cenveo, and HP's other customers operate digital presses manufactured by HP, including without

'028 Patent, Claim 1	
	limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description code specification, the page description code specification defining at least one data area, and the page description code further defining a graphics state including at least one attribute which controls the appearance of data in the data area;	HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in step (b) below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page

'028 Patent, Claim 1	
	includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'028 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p>
</td></tr>
<tr><td>(b) interpreting the page description code specification, and during the interpretation step, identifying the data area defined by the page description</td><td>HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production</td></tr>
</table>

'028 Patent, Claim 1	
code specification;	Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

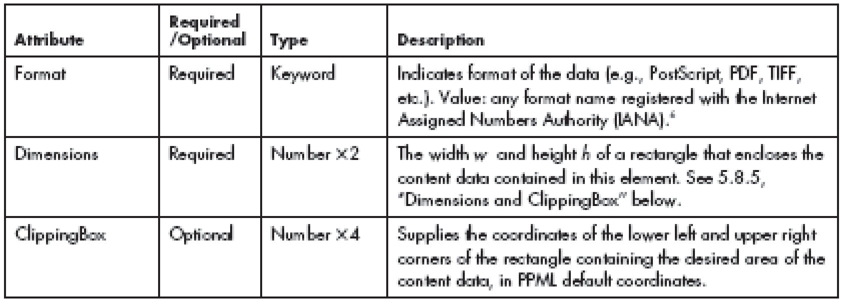
'028 Patent, Claim 1	
	types.
(c) upon the identification of the data area in step (b), applying the graphics state corresponding to the data area to a set of alphanumeric characters so as to generate a plurality of character bit maps;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to characters associated with the variable data areas. The appearance information is applied to the characters by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.

‘028 Patent, Claim 1

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate

'028 Patent, Claim 1	
	image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. RIP software applies the graphics state as part of generating character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) storing the plurality of character bit maps;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to store character bitmaps. The character bitmaps are stored by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software stores the character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate

'028 Patent, Claim 1	
	the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) retrieving a variable data item from a plurality of variable data items;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

'028 Patent, Claim 1	
	types.
(f) associating the variable data item with the plurality of character bit maps;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to associate variable data items with the character bitmaps. The variable data items associated with character bitmaps are identified by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software necessarily associates the character bitmaps for each character in the respective variable data areas. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a variable data bit map for the variable data using the character bit maps; and	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to generate variable data bitmaps. The variable data bitmaps are generated by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and

'028 Patent, Claim 1	
	PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(h) repeating steps (e) through (g) for remaining variable data items in the plurality of variable data items, whereby the stored character bit maps are used repeatedly to generate a plurality of variable data bit maps.	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to use the stored character bitmaps for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the stored character bitmaps and the template bitmap are reused for each instance of the document. As discussed above, RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

‘028 Patent, Claim 1	
	types.

‘028 Patent, Claim 2	
2. The computer implemented method of claim 1, wherein the page description code specification represents a template and includes a static data area, and the computer implemented method further comprises the steps of:	The elements of claim 1 are described in the chart above. HP, Cenveo, and HP’s other customers operate software tools as part of a process by which HP, Cenveo, and HP’s other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template and includes a static data area, as described further in the “executing” step below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP’s digital presses such as the ones operated by HP, Cenveo, and HP’s other customers. Refs. [3]-[9].
executing portions of the page description code specification corresponding to the static data area to generate a template bit map; and	HP, Cenveo, and HP’s other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP’s other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor (“RIP”) software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP’s print servers or digital front end computers. HP, Cenveo, and HP’s other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template

‘028 Patent, Claim 2	
	bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.

'028 Patent, Claim 2	
merging each of the plurality of the variable data bit maps into clean copies of the template bit map to create a plurality of merged bit maps.	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5.

'028 Patent, Claim 4	
4. A computer implemented method for generating a reusable template bit map suitable for high-speed variable printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses.

'028 Patent, Claim 4	
	HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '028 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the '028 patent and of the technology claimed within the '028 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '028 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '028 patent. HP, Cenveo, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Cenveo, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Cenveo, and HP's other customers operate digital presses manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the "executing" step below.

'028 Patent, Claim 4	
generating a page description code specification, the page description code specification defining at least one variable data area and at least one static data area;	HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further in the "executing" step below. Each of these files further defines at least one variable data area, as described further in the "identifying" step below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet, and the HP SmartSTream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. To the extent that third-parties, such as Cenveo's customers and/or their print media agents, perform the step of generating these files, Cenveo and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Cenveo and HP's other customers enter contracts with these third parties through which Cenveo and HP's other customers enforce the obligations that it imposes upon third-parties. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description code specification, and during the interpreting step,	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image

'028 Patent, Claim 4	
	processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers.
generating a bitmap of the static data area and adding the bitmap of the static data area to a template bitmap;	HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the

'028 Patent, Claim 4	
	system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'028 Patent, Claim 4	
identifying the variable data area, and	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
responsive to the identification	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs,

'028 Patent, Claim 4	
of the variable data, not adding a bitmap of the variable data area to the template bitmap; and	and Datasets, and therefore does not include a bitmap of the variable data area. Adding a bitmap of the variable data area to the template bitmap would prevent reuse of the static bitmap. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
saving the template bitmap, whereby copies of the template bitmap can be continuously accessed to create a plurality of variable data bitmaps.	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and

'028 Patent, Claim 4	
	can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

U.S. Patent No. 7,274,479 ("the '479 patent")

'479 Patent, Claim 9	
9. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '479 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the '479 patent and of the technology claimed within the '479 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '479 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '479 patent. HP, Cenveo, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Cenveo, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Cenveo, and HP's other customers operate digital presses manufactured by HP, including without

'479 Patent, Claim 9	
	limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a print specification, the print specification defining at least one variable data area and at least one static data area;	HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a static data area, as described further below. Each of these files further defines at least one variable data area, as described further in element (b) below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5.

'479 Patent, Claim 9	
	In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) providing a plurality of variable data items;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers.

'479 Patent, Claim 9	
	For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) identifying the variable data area;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Cenveo, and HP's other customers use such print servers or digital front ends to process VDP

'479 Patent, Claim 9	
	files including one or more of PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data, as described above in element (a). The VDP file defines variable data areas based on the surrounding tags of the data element.
(d) associating a graphic state with the variable data area, the graphic state including at least one attribute controlling the appearance of items to be printed in the variable data area;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to associate appearance information found in the VDP file to the corresponding variable data. The VDP file includes information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the

‘479 Patent, Claim 9	

Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data.

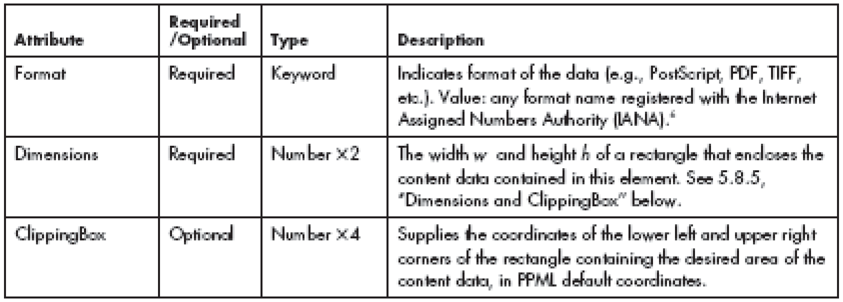
The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.

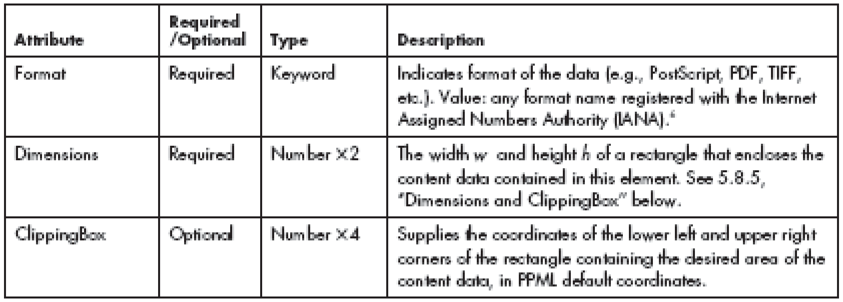


Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

'479 Patent, Claim 9	
	Ref. [11] at 28; Ref. [12] at 40. In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. As described above in element (b), variable data may be stored within the VDP file or in one or more separate files. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) retrieving a variable data item from the plurality of variable data items;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the

‘479 Patent, Claim 9	
	RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., “<xsl: value-of select=’name’/>” points to a database entry for the “name” element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Cenveo, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) generating a bitmap for the variable item, the generating step including a step of applying the graphic state associated with the variable data area to the variable data item; and	HP, Cenveo, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP’s print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format,

'479 Patent, Claim 9	
	dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p>
</td></tr>
<tr><td>(g) repeating steps (e) and (f) for remaining variable data items in the plurality of variable data items, whereby the graphic state associated with the</td><td>HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document.</td></tr>
</table>

'479 Patent, Claim 9	
variable data area is applied repeatedly to generate a plurality of variable data bitmaps.	The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of

'479 Patent, Claim 9	
	variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 10	
10. The method of claim 9, wherein the graphic state associated with the variable data area is defined within the print specification.	Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data, as discussed above with respect to element (d) of claim 9 of the '479 Patent. The appearance information may be defined within the print specification either by referencing an external file or by providing the appearance information directly within the VDP file.

'479 Patent, Claim 15	
15. The method of claim 9, wherein the variable data area and the static data area are defined, at least in part, by page description language commands.	As described for claim 9 of the '479 Patent, the VDP file defines static and variable data areas based on the surrounding tags of the data element. VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects each incorporate page description language commands. Each of these files is a page description language file, and the tags and commands included in each of these files are therefore page description language commands. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing, as described in elements (a) and (b) of claim 9. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 17	
17. The method of claim 9, further comprising a step of caching a representation of the static data area, whereby the cached representation of the static data area is available for merging with the variable data bitmaps to generate merged documents.	A static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based

'479 Patent, Claim 17	
	on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5.

'479 Patent, Claim 18	
18. The method of claim 17, wherein the cached representation of the static data area is a bitmap representation.	The cached representation of the static data area is a bitmap to avoid the redundant burden of the system to continually compute the contents appearance, as discussed above for claim 17 of the '479 Patent. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves

'479 Patent, Claim 18	
	downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 19	
19. The method of claim 9, wherein: the plurality of data items are associated with a field name; and	As described for claim 9 of the '479 Patent, each field retrieved from a variable data record is matched to a corresponding named variable data area defined within the VDP file.
the step of identifying a variable data area includes the step of detecting, in the print specification, a character string associated with the variable data area that matches the field name associated with the plurality of data items.	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, in PDF/VT files, document part metadata provides field name information for variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data. Ref. [12] at 42-43.

'479 Patent, Claim 19	
	In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.

U.S. Patent No. 7,333,233 ("the '233 patent")

'233 Patent, Claim 12	
12. A computer implemented method for generating a static bitmap suitable for high-speed variable printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '233 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the '233 patent and of the technology claimed within the '233 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '233 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '233 patent. HP, Cenveo, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Cenveo, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Cenveo, and HP's other customers operate digital presses manufactured by HP, including without

'233 Patent, Claim 12	
	limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below, and in the "interpreting" step. Each of these files further defines at least one variable data area, as described further below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C.

'233 Patent, Claim 12	
	In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained

'233 Patent, Claim 12	
	within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description language file, and during the interpreting step, generating a static bitmap of the static data area;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo,

'233 Patent, Claim 12	
	or Esko installed on HP's print servers or digital front end computers. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. The VDP file defines static data areas based on the surrounding tags of the data element. The type

‘233 Patent, Claim 12	
	of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file “associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to “content packages” that “include any static content in the file (text and image page objects, for instance).” Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
and saving the static bitmap, whereby the saved static bitmap is used repeatedly in the generation of a plurality of documents, each of which contains the static bitmap and a variable data bitmap.	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, “PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these

'233 Patent, Claim 12	
	items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it

'233 Patent, Claim 12	
	takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving, retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data

‘233 Patent, Claim 12	
	area present in a PPMLT file, an embedded XSLT “for-each” command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. HP, Cenveo, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

‘233 Patent, Claim 14	
14. A computer implemented method for generating a plurality of bitmaps suitable for high-speed printing, comprising the steps of:	Defendant Hewlett-Packard (“HP”), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print (“VDP”) jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Cenveo’s facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP’s Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Cenveo and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the ‘233 patent, and, on information and belief, HP has supplied related training and support materials and services to Cenveo and other HP customers. Despite its awareness of the ‘233 patent and of the technology claimed within the ‘233 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the ‘233 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the ‘233 patent.

'233 Patent, Claim 14	
	HP, Cenveo, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Cenveo, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Cenveo, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Cenveo, and HP's other customers operate digital presses manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	HP, Cenveo, and HP's other customers operate software tools as part of a process by which HP, Cenveo, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below. Each of these files further defines at least one variable data area, as described further below. HP provides at least some software tools that are part of a process by which Cenveo and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Cenveo, and HP's other customers. Refs. [3]-[9]. The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing.

'233 Patent, Claim 14	
	For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point

'233 Patent, Claim 14	
	to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'233 Patent, Claim 14	
(b) providing a merge file including a plurality of variable data items;	The VDP files can use variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which stores data either internally or externally. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. Ref. [17] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) processing the page description language file, and during the processing step, generating a static bitmap of the static data area and associating the variable data area with the plurality of variable data items; and	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Cenveo, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Cenveo, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP

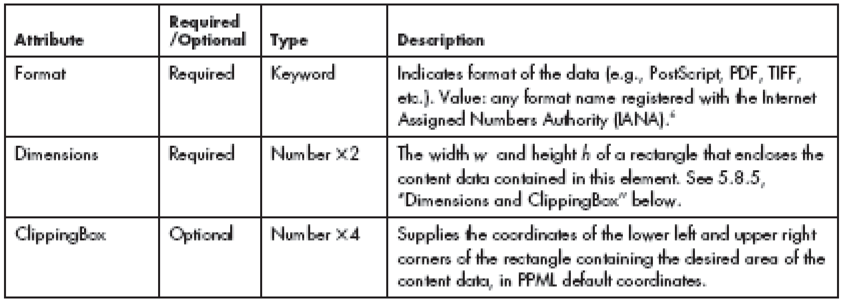
'233 Patent, Claim 14	
	file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The

'233 Patent, Claim 14	
	type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to associate variable data areas with variable data items. For example, in PDF/VT files, document part metadata provides field name information for variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data. Ref. [12] at 42-43.

'233 Patent, Claim 14	
	In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.
(d) saving the static bitmap;	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13.

'233 Patent, Claim 14	
	In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) generating a first variable data bitmap of a first one of the variable data items utilizing a graphics state associated with the variable data area;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g.,

'233 Patent, Claim 14	
	PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'233 Patent, Claim 14</th><th></th></tr>
<tr><td></td><td>


<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).⁴</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width <i>w</i> and height <i>h</i> of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(f) merging the first variable data bitmap with a copy of the static bitmap to produce a first output bitmap;</td><td>
HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. <i>See</i> Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap.

For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with
</td></tr>
</table>

'233 Patent, Claim 14	
	the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a next variable data bitmap of a next one of the variable data items utilizing a graphics state associated with the variable data area;	HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document, according to the contentions with respect to element (e). Appearance information is reused for each instance of the document. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. HP, Cenveo, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused

'233 Patent, Claim 14	
	for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data

'233 Patent, Claim 14	
	causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Cenveo, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
and (h) merging the next variable data bitmap with a copy of the static bitmap to produce a next output bitmap;	The print server or by a digital front end merges the variable data bitmaps with the template bitmap according to the contentions with respect to element (f). The appearance information and the template bitmap are reused for each instance of the document. The template bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. The template bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the

'233 Patent, Claim 14	
	system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2.
and (i) repeating steps (g) (h) for remaining variable data items in the plurality of variable data items.	The activities performed for steps (g) and (h) are repeated for each remaining variable data item in the plurality of data items.

Exhibit 6
to Maloney Declaration

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

Industrial Print Technologies, LLC, Plaintiff, v. Fort Dearborn Company and Hewlett-Packard Company, Defendants.	Civil Action No. 3:15-cv-01195-M The Honorable Barbara M.G. Lynn THIS DOCUMENT RELATES TO CIVIL ACTION NO. 3:15-cv-01195-M

PLAINTIFF INDUSTRIAL PRINT TECHNOLOGIES' DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS

In accordance with Miscellaneous Order No. 62 ¶¶ 3-1 and 3-2, and the Court's Order during the April 20, 2015 telephonic hearing, plaintiff Industrial Print Technologies LLC ("IPT") submits its Disclosure of Asserted Claims and Infringement Contentions as to Defendants Fort Dearborn Company ("Fort Dearborn") and Hewlett-Packard Company ("HP") (collectively, "Defendants").

1. Right to Supplement

IPT bases these disclosures on its current knowledge, understanding and belief as to the facts and information available to it as of the date of these disclosures. Discovery in this case has not yet opened, and IPT has not yet completed its investigation, collection of information, discovery or analysis related to this action. Accordingly, IPT reserves the right to supplement, amend or modify the information contained herein and to use and introduce such information and any subsequently-identified information at trial. In particular, IPT reserves its right to amend and supplement its identification of asserted claims and modify its identification of accused products and instrumentalities. Additionally, as further discovery is taken, and additional details are provided regarding Defendants' activities, IPT may seek to amend and/or supplement. IPT also reserves its right to supplement its disclosure of documents based upon further investigation and discovery.

These disclosures are based at least in part upon IPT's present understanding of the meaning and scope of the claims of the patents-in-suit, in the absence of additional claim construction proceedings or discovery. IPT reserves the right to seek leave to supplement or amend these disclosures if its understanding of the claims changes, including if the Court construes them.

2. Asserted Claims

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(1), based on information presently available to IPT, IPT states that Defendants infringe:

U.S. Patent No. 5,729,665 ("the '665 patent"), claims 1, 12, 13, and 20;

U.S. Patent No. 5,937,153 ("the '153 patent"), claims 1, 3-5, and 6;

U.S. Patent No. 6,381,028 ("the '028 patent"), claims 1, 2, and 4;

U.S. Patent No. 7,274,479 ("the '479 patent"), claims 9, 10, 15, and 17-19; and

U.S. Patent No. 7,333,233 ("the '233 patent"), claims 12 and 14.

IPT reserves the right to assert additional claims against Defendants based upon results of discovery and further investigation.

3. Accused Instrumentalities and Comparison To Asserted Claims

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(2), based on information presently available to IPT, Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has been and is engaged in infringing activities using variable data enabled high-speed printing presses supplied by Defendant HP. Specifically, Fort Dearborn has been and is engaged in infringing the asserted method claims under 35 U.S.C. §

271(a) through its use of HP's high-speed printing presses that process variable data print ("VDP") jobs, including at least HP Indigo Digital Presses (including for example at least Indigo WS4000 and Indigo WS4050 presses) and by selling and/or offering to sell related variable data printing services to its customers within the United States. Fort Dearborn has also been and is infringing under 35 U.S.C. § 271(g) by selling and/or offering to sell print materials containing variable data which are made using methods covered by the patented methods to its customers within the United States.

To the extent that any steps of the methods covered by the asserted patent claims are performed by third-parties, such as Fort Dearborn's customers and/or their print media agents, Plaintiff alleges that Fort Dearborn is liable for direct infringement because it directs and controls any such third-party steps, including, for example, by dictating the manner by which the third-parties must supply data to enable variable data print jobs to be run on Fort Dearborn's variable data enabled high-speed printing presses, such that Fort Dearborn is jointly and severally and/or vicariously liable for any acts performed by such third-parties on behalf of Fort Dearborn. Further, upon information and belief, Fort Dearborn enters contracts with these third parties, through which Fort Dearborn enforces the obligations that it imposes upon third-parties.

HP directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe under 35 U.S.C. § 271(a) by setting up and running VDP jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn facilities, including at least by operating Indigo Digital Presses. HP, directly and/or through its subsidiaries, affiliates, agents, and/or business partners has also induced and continues to induce Fort Dearborn and other HP customers to commit direct infringement of the asserted claims pursuant to 35 U.S.C. § 271(b) by one or more of supplying, offering for sale and selling at least its Indigo Digital Presses, which were designed and intended to practice methods covered by the asserted claims. HP has also supplied related training and support materials and services. Despite its awareness of the asserted claims and of the technology claimed within the asserted claims, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the asserted patent claims and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the asserted patent claims.

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(3), IPT provides the following charts, attached as Appendices A and B, that identify specifically where each element and/or step of each asserted claim is found within the Defendants' infringing methods and systems. IPT reserves the right to amend, supplement and modify its contentions and charts based on additional information identified through discovery.

4. Literal and Equivalents Infringement

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(4), as supported and explained in the attached Appendices, IPT currently believes that each of the elements of each of the asserted claims is met literally, and if any claim or claim limitation is not met literally, then it is met under the doctrine of equivalents.

It is expected that the same facts upon which IPT's literal infringement claim is based will also form the basis of IPT's doctrine of equivalents claim, as any differences between the limitations of the asserted claims and the accused products are insubstantial. With respect to the doctrine of equivalents, however, as Defendants have not yet provided details of their non-infringement positions, IPT reserves the right to present further facts to support an assertion of infringement under the doctrine of equivalents.

5. Priority Date

In accordance with Miscellaneous Order No. 62 ¶ 3-1(a)(5), IPT alleges that each asserted claim of all five asserted patents is entitled to a priority date at least as early as January 18, 1995, which is the filing date of the ‘665 patent, to which the other asserted patents also claim priority.

The subject matter of the asserted claims of the asserted patents was conceived of prior to the filing of the application that became the ‘665 patent.

IPT believes that the subject matter of the asserted claims was conceived of at least as early as 1988, and no later than 1989. The subject matter of the asserted claims was then diligently reduced to practice through the first operating prototype that was completed on or about February 10, 1994. IPT thus contends that the claims are entitled to an invention date during 1989. There was constructive reduction to practice on January 18, 1995. To the extent that further investigation and discovery permits a more specific invention date to be confirmed, IPT will update its disclosures as appropriate.

6. Documents

7. IPT has made a reasonable investigation for documents identified in Miscellaneous Order No. 62 ¶ 3-2. Such non-privileged documents are being produced herewith**.**

In accordance with Miscellaneous Order No. 62 ¶ 3-2(a), IPT’s documents corresponding to ¶ 3-2(a)(1) include at least those numbered:

TES002976-TES002980, TES004201-TES004202, TES004207-TES004209, TES004210-TES004211, TES004212-TES004245, TES004250-TES004278, TES004279-TES004280, TES004281-TES004282, TES004283-TES004284, TES004320-TES004324, TES004325-TES004330, TES004331-TES004333, TES004415-TES004415, TES004416-TES004416, TES004812-TES004812, TES004813-TES004814, TES004822-TES004827, TES004828-TES004833, TES004834-TES004838, TES004843-TES004844, TES004847-TES004848, TES004858-TES004860, TES004861-TES004863, TES004864-TES004866, TES004867-TES004869, TES005505-TES005521, TES005522-TES005527, TES009900-

TES010246, TES011201-TES011202, TES013273-TES013304, TES013477-TES013478, TES015782-TES015786, TES018684-TES018720, TES036025-TES036138, TES107224-TES107234, TES108742-TES108775, TES108776-TES108798, TES108799-TES108821, TES237440-TES237442, TES240475-TES240608.

IPT's documents corresponding to ¶ 3-2(a)(2) include at least those numbered:

TES002250-TES002253, TES002269-TES002271, TES002305-TES002422, TES002870-TES002873, TES003038-TES003047, TES003856-TES003993, TES003998-TES004029, TES004077-TES004078, TES004083-TES004100, TES004104-TES004104, TES004119-TES004163, TES004184-TES004197, TES004203-TES004203, TES004207-TES004245, TES004250- ES004284, TES004286-TES004291, TES004293-TES004303, TES004305-TES004310, TES004320-TES004333, TES004365-TES004398, TES004403-TES004405, TES004409-TES004409, TES004417-TES004420, TES004445-TES004455, TES004478-TES004478, TES004480-TES004480, TES004481-TES004487, TES004489-TES004523, TES004525-TES004545, TES004551-TES004551, TES004579-TES004607, TES004614-TES004665, TES004669- TES004717, TES004724-TES004729, TES004731-TES004798, TES004812-TES004814, TES004816-TES004871, TES004880-TES005020, TES005028-TES005099, TES005107-TES005290, TES005299-TES005304, TES005306-TES005350, TES005352-TES005380, TES005382-TES005383, TES005385-TES005437, TES005457-TES005458, TES005460-TES005470, TES005505-TES005521, TES005528, TES005532-TES005540, TES005564-TES005591, TES005598-TES005607, TES005609-TES005645, TES005672-TES005680, TES006504-TES006653, TES006695-TES006695, TES006723-TES006724, TES006816-TES006846, TES007208-TES007223, TES008359-TES008448, TES008463-TES008584, TES008614-TES008620, TES008650-TES008680, TES008691-TES008695, TES009442-TES009503, TES009525-TES009537, TES009595, TES009659-TES009662 TES009848-TES009899, TES011141-TES011200, TES011203-TES011303, TES011310-TES011372, TES011608-TES011669, TES011817-TES011820, TES011823-TES011986, TES012004-TES012014, TES012040-TES012054, TES012290-TES012354, TES013081-TES013174, TES013273-TES013304, TES014021-TES014151, TES014190-TES015304, TES015787-TES015799, TES015810-TES015813, TES016292-TES016334, TES018613-TES018623, TES018626-TES018679, TES019295-TES019351, TES019356-TES019379, TES022843-TES022853, TES023472-TES023476, TES025611-TES025624, TES025626-TES025679, TES032626-TES032657, TES032664-TES032695, TES038176-TES038282, TES038419-TES038585, TES038623-TES038694, TES038829-TES039181, TES040237-TES040526, TES040784-TES041088, TES041343-TES041422, TES047510-TES047514, TES100247-TES100251, TES100286-TES100287, TES100293-TES100326, TES100580-TES100580, TES100604-TES100610, TES107224-TES107234, TES274326-TES274326, TES279177-TES279177, TES280365-TES280365, TES280374-TES280374, TES281386-TES281386, TES281730-TES281730, TES281739-TES281739, TES281747-TES281747.

IPT's documents corresponding to ¶ 3-2(a)(3) include at least those numbered:

TES336688-TES336813, TES337205-TES337279, TES337507-TES337622, TES337623-TES337713, TES338116-TES338285, TES338286-TES338324, TES340745-

TES342864, TES342865-TES344969, TES344970-TES347044, TES347045-TES349151, TES349455-TES352270, TES352271-TES355288.

Date: May 11, 2015

/s/ David A. Gosse
Timothy P. Maloney (IL 6216483)
Alison Aubry Richards (IL 6285669)
Nicole L. Little (IL 6297047)
David A. Gosse (IL 6299892)
FITCH, EVEN, TABIN & FLANNERY LLP
120 South LaSalle Street, Suite 1600
Chicago, Illinois 60603
Telephone: (312) 577-7000
Facsimile: (312) 577-7007

Steven C. Schroer (IL 6250991)
scschr@fitcheven.com
FITCH, EVEN, TABIN & FLANNERY LLP
1942 Broadway, Suite 213
Boulder, CO 80302
Telephone: 303.402.6966
Facsimile: 303.402.6970

Counsel for Plaintiff

CERTIFICATE OF SERVICE

The undersigned certifies that a copy of the above document PLAINTIFF IPT'S DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS and exhibits was sent by email to the counsel listed below on this May 11, 2015:

Edward R. Reines
edward.reines@weil.com
Sonal N. Mehta
sonal.mehta@weil.com
Evan N. Budaj
evan.budaj@weil.com
WEIL, GOTSHAL & MANGES LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

Audrey L. Maness
WEIL, GOTSHAL & MANGES LLP
700 Louisiana, Suite 1700
Houston, TX 77002
Telephone: (713) 546-5317
Facsimile: (713) 224-9511

Melissa Richards Smith
melissa@gillamsmithlaw.com
GILLAM & SMITH, LLP
303 South Washington Avenue
Marshall, TX 75670
Telephone: (903) 934-8450
Facsimile: (903) 934-9257

Counsel for Defendants Fort Dearborn Co. and Hewlett-Packard Co.

/s/ David A. Gosse
David A. Gosse
Attorney for Plaintiff

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. These contentions are based solely on publicly available information relating to exemplary variable data ("VDP") files and raster image processor (RIP") software and press control software used on HP's presses. Discovery in this case has not yet begun, and the charts below do not reflect any information produced by defendants Fort Dearborn Company and Hewlett-Packard Company. Other VDP files, RIP software and press control software may be identified through discovery as being used by defendants. IPT reserves the right to support its contentions with additional material produced by the defendants or subsequently identified by IPT.

[1] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf
[2] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf
[3] HP T200 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T200_Data_Sheet.pdf
[4] HP T300 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T300_Data_Sheet.pdf
[5] HP T350 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T350_Data_Sheet.pdf
[6] HP T400 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T400_Data_Sheet.pdf
[7] HP Indigo w3250 Data Sheet, available at http://h10010.www1.hp.com/wwpc/pscmisc/vac/us/product_pdfs/90566.pdf
[8] HP Indigo 5600 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2012/HPPredrupa12/HP_Indigo_5600.pdf
[9] HP Indigo 7500 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2010/IPEX2010/HP_Indigo_7500_DS.PDF
[10] PPML Template, available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html
[11] PPML Specification v1.5, available at http://www.standards.podi.org/ppml/specification.html
[12] PPML Specification v2.1, available at http://www.standards.podi.org/ppml/specification.html

[13] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf
[14] HP Indigo Yours Truly Designer 7 User Guide
[15] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.
[16] Adobe Systems Inc., PDF Reference 5th Ed., v. 1.6, available at http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference16.pdf
[17] ISO 16612-2:2010, available at http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=46428
[18] Global Graphics, Do PDF/VT Right, available at http://www.globalgraphics.com/doPDFVTright/

U.S. Patent No. 5,729,665 ("the '665 patent")

'665 Patent, Claim 1	
1. A method for generating multiple bit maps suitable for high-speed printing or plate-making comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code representing a template, said page description code defining at least one	Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in element (b) below. Each of these files further

'665 Patent, Claim 1	
variable data area and said page description code further defining a graphics state corresponding to said variable data area, said graphics state including at least one attribute which controls the appearance of variable data in said variable data area;	defines at least one variable data area, as described further in element (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn enters contracts with these third parties through which Fort Dearborn enforces the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper

‘665 Patent, Claim 1	
	right corners of the rectangle containing the desired area of the content data.

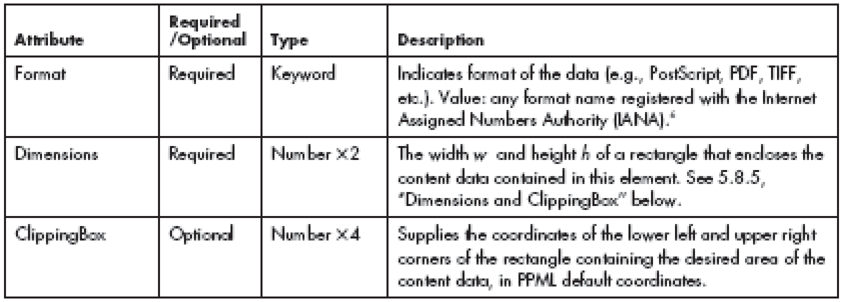
The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

'665 Patent, Claim 1	
	Ref. [11] at 28; Ref. [12] at 40. In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) executing said page description code to generate a bit map of said template, and during said execution, identifying said variable data area defined by said page description code and reserving said graphics state corresponding to said variable data area upon said identification;	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying

'665 Patent, Claim 1	
	reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The

'665 Patent, Claim 1	
	type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of

‘665 Patent, Claim 1	
	PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) retrieving variable data;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., “<xsl: value-of select=’name’/>” points to a database entry for the “name” element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4.

'665 Patent, Claim 1	
	Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) associating said variable data with said graphics state corresponding to said variable data area;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to associate the appearance information found in the VDP file to the corresponding variable data. As described above, variable data may be stored within the VDP file or in one or more separate files. The RIP software associates the variable data with the appearance information defined for the variable data area, as described further in element (e) below.
(e) applying said graphics state corresponding to said variable data area to said variable data to generate a variable data bit map; and	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an

'665 Patent, Claim 1	

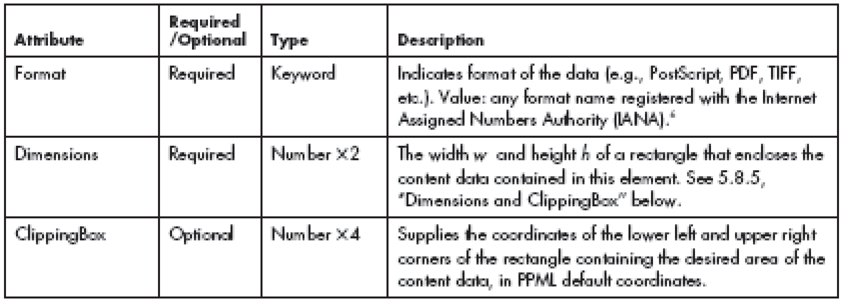
Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34.

"The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.

"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.

"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.

"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The

'665 Patent, Claim 1	
	appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) merging said variable data bit map with said bit map of said template;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of

'665 Patent, Claim 1	
	content for a given printed page. Ref. [15] at 4-5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
wherein said graphics state corresponding to said variable data area is applied repeatedly to variable data to generate a multitude of variable data bit maps without the need to repeat said executing step (b).	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15.

'665 Patent, Claim 1	
	PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

'665 Patent, Claim 12	
12. The method of claim 1 wherein said reserving, retrieving, associated, applying, and merging steps are repeated for each variable data area defined by said page description code.	VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving,

'665 Patent, Claim 12	
	retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'665 Patent, Claim 13	
13. The method of claim 12 wherein said reserving, retrieving, associating, applying and merging steps are activated by a control task running in a printer controller, and wherein said control task interrupts said page description code execution upon identifying a predetermined command in said page description code.	As described above, the steps of reserving, retrieving, associating, applying and merging are all activated and monitored by a control task running in a dedicated print server or digital front end associated with the press. Each of the respective print servers or digital front ends runs RIP software such as the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The control task interrupts said page description code execution upon identifying a predetermined command in said page description code to enable other operations to be performed.

'665 Patent, Claim 20	

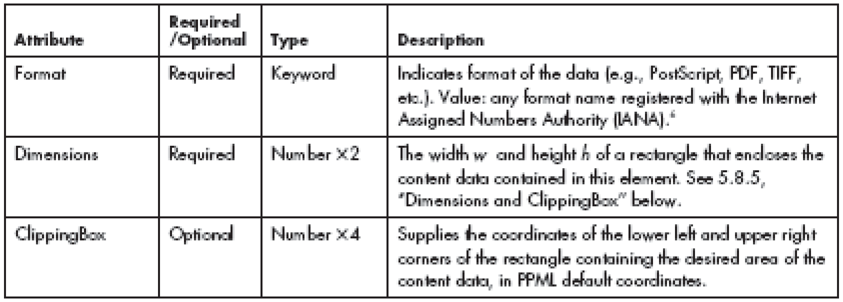
‘665 Patent, Claim 20	
20. A method for generating a plurality of bit maps suitable for high-speed printing or plate-making from a page description code representing a template and defining at least one variable data area, and from a merge file containing a plurality of data records of at least one variable data field type, the method comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the "executing a control task" step below. Each of these files further defines at least one variable data area, as described further in the "executing a control task" step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn enters contracts with these third parties through which Fort Dearborn enforces the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data.

'665 Patent, Claim 20	
	For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and

‘665 Patent, Claim 20	

orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to

'665 Patent, Claim 20	
	retrieve variable data elements stored in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be referenced from outside of the PPMLT file itself. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a page description code interpretive program, said interpretive program generates graphics states for each data area defined by said page description code;	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global

‘665 Patent, Claim 20	
	Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The RIP software necessarily includes a module or other discrete software component that interprets VDP files, including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper

‘665 Patent, Claim 20

right corners of the rectangle containing the desired area of the content data.

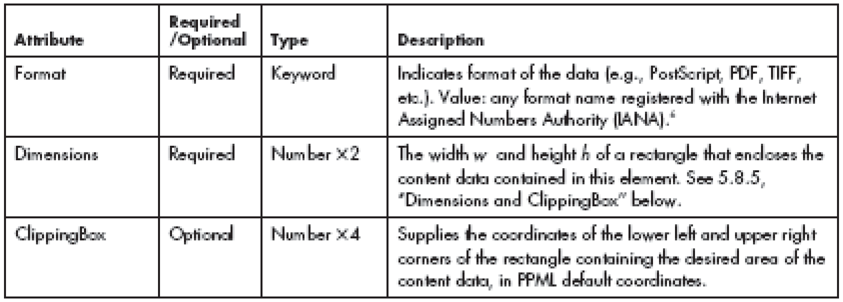
The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

'665 Patent, Claim 20	
	In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a control task in conjunction with said interpretive program, said control task identifies said variable data area defined by said page description code and reserves said graphics states generated by said interpretive program for said variable data area, said control task generates a template bit map defined by said page description code, and after the completion of said interpretive program, said control task saves said template bit map in memory; and	As described above, Fort Dearborn runs software on dedicated print servers or digital front ends. RIP software identifies variable data areas defined in VDP files. The RIP software necessarily includes one or more module or other discrete software component that identifies variable data areas, reserves graphics states, and generates one or more template bitmap, and saves one or more template bitmap in memory. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet

'665 Patent, Claim 20	
	presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata

‘665 Patent, Claim 20	
	that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag “associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to “content packages” that “include any static content in the file (text and image page objects, for instance).” Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT

'665 Patent, Claim 20	
	files define stored channels that include scaling and rotation parameters for each element. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a merge task upon completion of said interpretive program, said merge task generates variable data bit maps for said data records in said merge file by applying said reserved graphics states to said data records, and said merge task merges said variable data bit maps with a separate copy of said template bit map to create the plurality of bit maps suitable for high-speed printing or plate making;	As described above, Fort Dearborn runs software on dedicated print servers or digital front ends. RIP software applies appearance information found in the VDP file to the corresponding variable data areas. The RIP software necessarily includes one or more module or other discrete software component that applies the appearance information to the corresponding variable data to generate a variable data bit map. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT,

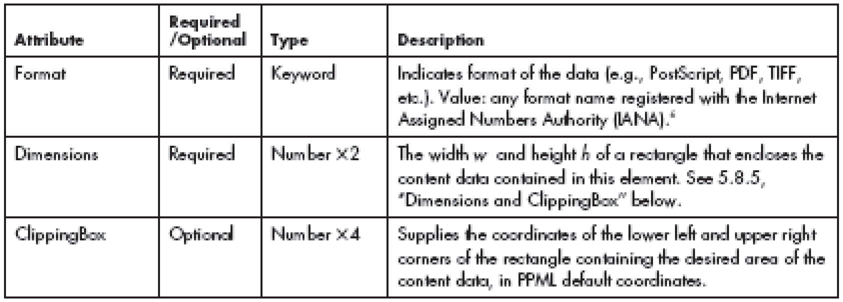
'665 Patent, Claim 20	

REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.

"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.

"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.

"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and

‘665 Patent, Claim 20	
	optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, “PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction.” Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define “channels” that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
whereby said reserved graphics states are applied repeatedly to said data records to generate said variable data bit maps for	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but

‘665 Patent, Claim 20	
said data records without the need to repeat said steps of executing a page description code interpretive program and executing a control task in conjunction with said interpretive program.	the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT “for-each” command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data

'665 Patent, Claim 20	
	causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

U.S. Patent No. 5,937,153 ("the '153 patent")

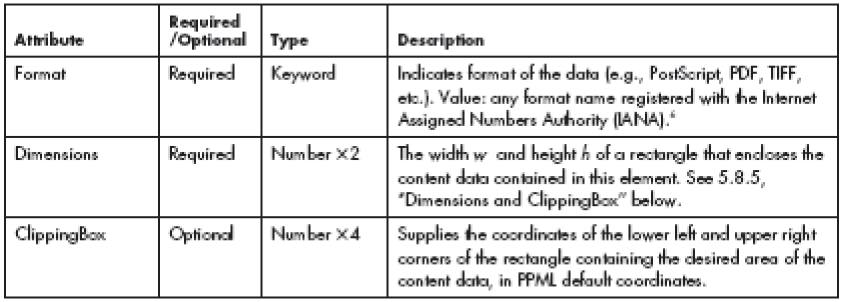
'153 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code specification, the page description code specification defining at least one data area to become variable, and the page description code further defining a graphics state corresponding to the data area, the graphics state including at least one attribute which controls the appearance of data in the data area;	Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in element (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartSTream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn enters contracts with these third parties through which Fort Dearborn enforces the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data.

‘153 Patent, Claim 1	
	For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and

‘153 Patent, Claim 1

orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions

'153 Patent, Claim 1	
	similar to those described above in these exemplary file types.
(b) interpreting the page description code specification, and during the interpretation, identifying the data area defined by the page description code specification;	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to

'153 Patent, Claim 1	
	identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) storing the graphics state corresponding to the data area upon the identification of the variable data area in step (b);	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) retrieving a variable data item from a plurality of variable data items;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends.

'153 Patent, Claim 1	
	In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) applying the stored graphics state to the variable data item to generate a variable data bit map; and	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of

‘153 Patent, Claim 1	

the content data.

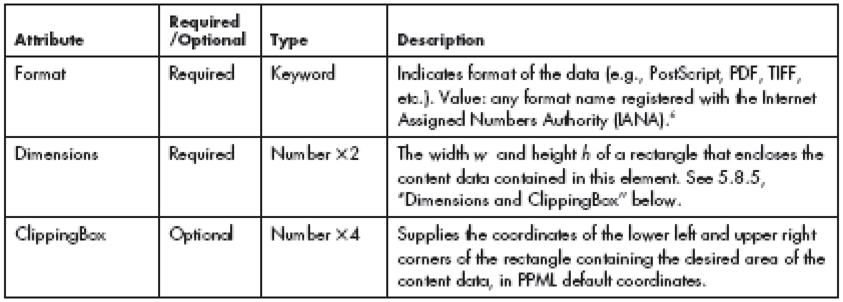
The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).⁴
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

'153 Patent, Claim 1	
	Ref. [11] at 28; Ref. [12] at 40. In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) repeating steps (d) and (e) for remaining variable data items in the plurality of variable data items, whereby the stored graphics state is applied repeatedly to generate a plurality of variable data bit maps.	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly,

'153 Patent, Claim 1	
	allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 3	
3. The computer implemented method of claim 1, wherein the page description code specification represents a	As described for claim 1 of the '153 patent, Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files. Each of these files represents a template. These VDP files use static data areas to quickly manage VDP jobs. PPML for example, performs

'153 Patent, Claim 3	
template and includes a static data area; and the computer implemented method further comprises the steps of:	more efficiently when the static data areas are defined in advance. Ref. [10] at 10.
executing portions of the page description code specification corresponding to the static data area to generate a template bit map;	Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3.

‘153 Patent, Claim 3	
	In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
storing the template bit map; and	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. “Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a

'153 Patent, Claim 3	
	picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
merging each of the plurality of the variable data bit maps into a clean copy of the template bit map to create a plurality of merged bit maps.	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data

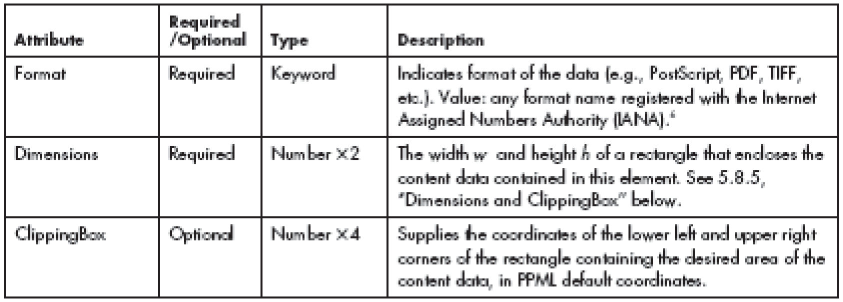
'153 Patent, Claim 3	
	which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 4	
4. The computer implemented method of claim 1, wherein the identifying step includes the step of detecting predefined characters within a text string defined in the page description code specification.	As described for claim 1 of the '153 patent, the controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files use objects denoted by the text "obj" to identify template and variable data areas. Further, the text "/XObject" denotes information in certain objects that will be reused. The RIP software may detect these characters or the RIP software may evaluate repetitive text within the PDF files to identify data areas. In PDF/VT, XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." [17] at § 6.7.3. For example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a

'153 Patent, Claim 4	
	document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. Each of these structures is associated with a predetermined characters within the JLYT file. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 5	
5. The computer implemented method of claim 1, wherein the attribute is a size attribute, a font attribute, a position attribute, an orientation attribute or a location attribute.	As described above, PDF, PDF/VT, PPML, PPMLT, and JLYT can each define appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be

'153 Patent, Claim 5	
	printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>‘153 Patent, Claim 5</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p>
</td></tr>
</table>

‘153 Patent, Claim 6	
6. A computer implemented method for processing a page description code specification comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services and resulting printed products to its customers. Fort Dearborn operates software capable

'153 Patent, Claim 6	
	of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further below. Each of these files further defines at least one variable data area, as described further in the "interpreting" step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only

'153 Patent, Claim 6	
	rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.
interpreting the page description code specification, and during the interpretation, identifying a data area defined by the page description code specification;	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital

'153 Patent, Claim 6	
	front end computers. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 6	
upon the identification of the data area, storing a graphics state set forth in the page description code specification which defines an attribute of how data is to appear in the data area; and	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
repeatedly retrieving data records from a plurality of data records and applying the stored graphics state to the data records to generate a plurality of bitmaps of the data records so that the bitmaps of the data records include the attribute.	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends.

'153 Patent, Claim 6	
	In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement

‘153 Patent, Claim 6	

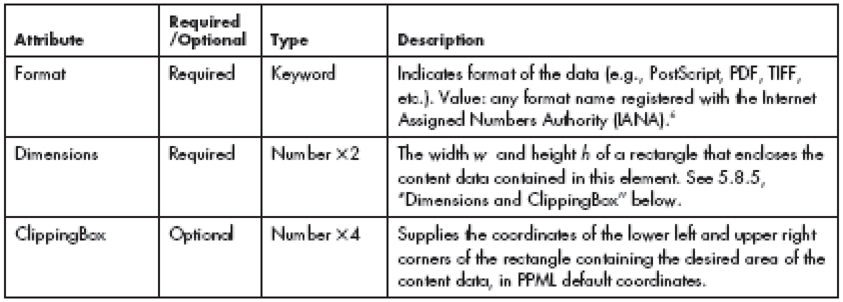
of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing,

'153 Patent, Claim 6	
	size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide

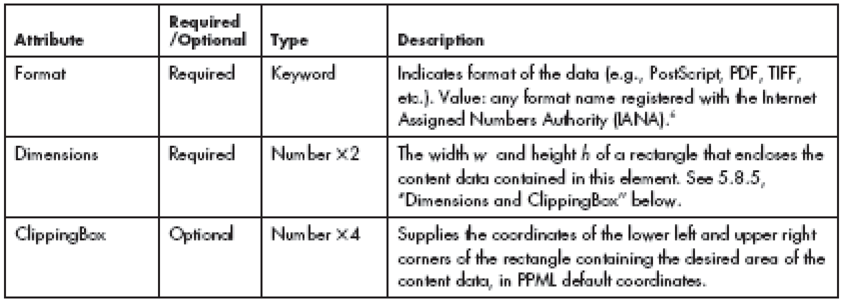
'153 Patent, Claim 6	
	information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

U.S. Patent No. 6,381,028 ("the '028 patent")

'028 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description code specification, the page description code specification defining at least one data area, and the page description code further defining a graphics state including at least one attribute which controls the appearance of data in the data area;	Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in step (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is

'028 Patent, Claim 1	
	controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data.

<table>
<tr><th>'028 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

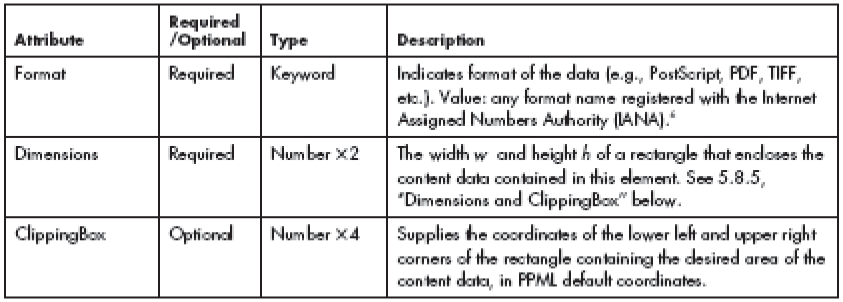
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</td></tr>
<tr><td>(b) interpreting the page description code specification, and during the interpretation</td><td>Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may</td></tr>
</table>

'028 Patent, Claim 1	
step, identifying the data area defined by the page description code specification;	comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5.

'028 Patent, Claim 1	
	Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) upon the identification of the data area in step (b), applying the graphics state corresponding to the data area to a set of alphanumeric characters so as to generate a plurality of character bit maps;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to characters associated with the variable data areas. The appearance information is applied to the characters by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT,

<table>
<tr><th>'028 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>
<p>REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36.</p>
<p>"The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.</p>
<p>"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.</p>
<p>"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.</p>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and</p>
</td></tr>
</table>

'028 Patent, Claim 1	
	optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. RIP software applies the graphics state as part of generating character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) storing the plurality of character bit maps;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to store character bitmaps. The character bitmaps are stored by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software stores the character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the

'028 Patent, Claim 1	
	example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) retrieving a variable data item from a plurality of variable data items;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'028 Patent, Claim 1	
(f) associating the variable data item with the plurality of character bit maps;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to associate variable data items with the character bitmaps. The variable data items associated with character bitmaps are identified by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software necessarily associates the character bitmaps for each character in the respective variable data areas. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a variable data bit map for the variable data using the character bit maps; and	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to generate variable data bitmaps. The variable data bitmaps are generated by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered

'028 Patent, Claim 1	
	characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(h) repeating steps (e) through (g) for remaining variable data items in the plurality of variable data items, whereby the stored character bit maps are used repeatedly to generate a plurality of variable data bit maps.	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to use the stored character bitmaps for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the stored character bitmaps and the template bitmap are reused for each instance of the document. As discussed above, RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'028 Patent, Claim 2	
2. The computer implemented method of claim 1, wherein the	The elements of claim 1 are described in the chart above. Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates,

'028 Patent, Claim 2	
page description code specification represents a template and includes a static data area, and the computer implemented method further comprises the steps of:	references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template and includes a static data area, as described further in the "executing" step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9].
executing portions of the page description code specification corresponding to the static data area to generate a template bit map; and	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only

'028 Patent, Claim 2	
	rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.
merging each of the plurality of the variable data bit maps into clean copies of the template bit map to create a plurality of merged bit maps.	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page.

'028 Patent, Claim 2	
	Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5.

'028 Patent, Claim 4	
4. A computer implemented method for generating a reusable template bit map suitable for high-speed variable printing, comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the "executing" step below.

'028 Patent, Claim 4	
generating a page description code specification, the page description code specification defining at least one variable data area and at least one static data area;	Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further in the "executing" step below. Each of these files further defines at least one variable data area, as described further in the "identifying" step below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartSTream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn directs and controls such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn enters contracts with these third parties through which Fort Dearborn enforces the obligations that it imposes upon third-parties. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description code specification, and during the interpreting step,	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers.
generating a bitmap of the static data area and adding the bitmap	Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types

‘028 Patent, Claim 4	
of the static data area to a template bitmap;	that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point

‘028 Patent, Claim 4	
	to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. “Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. In a further example, with respect to PPMLT documents, “PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer).” Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
identifying the variable data area, and	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing.

'028 Patent, Claim 4	
	For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
responsive to the identification of the variable data, not adding a bitmap of the variable data area to the template bitmap; and	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets, and therefore does not include a bitmap of the variable data area. Adding a bitmap of the variable data area to the template bitmap would prevent reuse of the static bitmap. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
saving the template bitmap,	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By

'028 Patent, Claim 4	
whereby copies of the template bitmap can be continuously accessed to create a plurality of variable data bitmaps.	identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server

'028 Patent, Claim 4	
	or digital front end. *See* Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

U.S. Patent No. 7,274,479 ("the '479 patent")

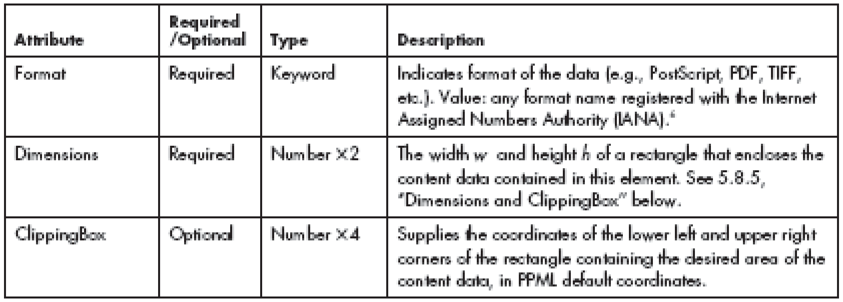
'479 Patent, Claim 9	
9. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing, comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a print specification, the print specification defining at least one variable data area and at least one static data area;	Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a static data area, as described further below. Each of these files further defines at least one variable data area, as described further in element (b) below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5.

'479 Patent, Claim 9	
	For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) providing a plurality of	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to

'479 Patent, Claim 9	
variable data items;	retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) identifying the variable data area;	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global

‘479 Patent, Claim 9	
	Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Fort Dearborn uses such print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data, as described above in element (a). The VDP file defines variable data areas based on the surrounding tags of the data element.
(d) associating a graphic state with the variable data area, the graphic state including at least one attribute controlling the appearance of items to be printed in the variable data area;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to associate appearance information found in the VDP file to the corresponding variable data. The VDP file includes information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The

'479 Patent, Claim 9	
	MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

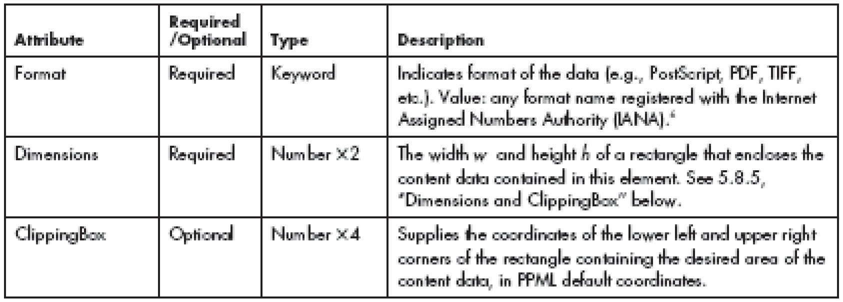
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

As described above in element (b), variable data may be stored within the VDP file or in one or more separate files. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field.

Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(e) retrieving a variable data item from the plurality of</td><td>Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The</td></tr>
</table>

'479 Patent, Claim 9	
variable data items;	variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) generating a bitmap for the variable item, the generating step including a step of applying the graphic state associated with the variable data area to the variable data item; and	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The

'479 Patent, Claim 9	
	graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.
In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.
Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</td></tr>
<tr><td>(g) repeating steps (e) and (f) for remaining variable data items in the plurality of variable data items, whereby the graphic state associated with the variable data area is applied</td><td>Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document.
The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data</td></tr>
</table>

'479 Patent, Claim 9	
repeatedly to generate a plurality of variable data bitmaps.	bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions

‘479 Patent, Claim 9	
	similar to those described above in these exemplary file types.

‘479 Patent, Claim 10	
10. The method of claim 9, wherein the graphic state associated with the variable data area is defined within the print specification.	Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data, as discussed above with respect to element (d) of claim 9 of the ‘479 Patent. The appearance information may be defined within the print specification either by referencing an external file or by providing the appearance information directly within the VDP file.

‘479 Patent, Claim 15	
15. The method of claim 9, wherein the variable data area and the static data area are defined, at least in part, by page description language commands.	As described for claim 9 of the ‘479 Patent, the VDP file defines static and variable data areas based on the surrounding tags of the data element. VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects each incorporate page description language commands. Each of these files is a page description language file, and the tags and commands included in each of these files are therefore page description language commands. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing, as described in elements (a) and (b) of claim 9. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

‘479 Patent, Claim 17	
17. The method of claim 9,	A static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets.

'479 Patent, Claim 17	
further comprising a step of caching a representation of the static data area, whereby the cached representation of the static data area is available for merging with the variable data bitmaps to generate merged documents.	By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server

‘479 Patent, Claim 17	
	or digital front end. *See* Ref. [15] at 5.

‘479 Patent, Claim 18	
18. The method of claim 17, wherein the cached representation of the static data area is a bitmap representation.	The cached representation of the static data area is a bitmap to avoid the redundant burden of the system to continually compute the contents appearance, as discussed above for claim 17 of the ‘479 Patent. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. “Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, “PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it

'479 Patent, Claim 18	
	takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 19	
19. The method of claim 9, wherein: the plurality of data items are associated with a field name; and	As described for claim 9 of the '479 Patent, each field retrieved from a variable data record is matched to a corresponding named variable data area defined within the VDP file.
the step of identifying a variable data area includes the step of detecting, in the print specification, a character string associated with the variable data area that matches the field name associated with the plurality of data items.	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, in PDF/VT files, document part metadata provides field name information for variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data. Ref. [12] at 42-43. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition,

'479 Patent, Claim 19	
	PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.

U.S. Patent No. 7,333,233 ("the '233 patent")

'233 Patent, Claim 12	
12. A computer implemented method for generating a static bitmap suitable for high-speed variable printing, comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below, and in the "interpreting" step. Each of these files further defines at least one variable data area, as described further below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas.

‘233 Patent, Claim 12	
	In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag “associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to “content packages” that “include any static content in the file (text and image page objects, for instance).” Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3.

'233 Patent, Claim 12	
	In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description language file, and during the interpreting step, generating a static bitmap of the static data area;	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global

'233 Patent, Claim 12	
	Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13.

'233 Patent, Claim 12	
	The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
and saving the static bitmap, whereby the saved static bitmap is used repeatedly in the generation of a plurality of documents, each of which contains the static bitmap and a variable data bitmap.	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these

'233 Patent, Claim 12	
	items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it

'233 Patent, Claim 12	
	takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving, retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data

'233 Patent, Claim 12	
	area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'233 Patent, Claim 14	
14. A computer implemented method for generating a plurality of bitmaps suitable for high-speed printing, comprising the steps of:	Defendant Fort Dearborn, directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to its customers. Fort Dearborn operates software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, Fort Dearborn operates presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, Fort Dearborn operates digital presses manufactured by HP, including: Indigo WS4000 and Indigo WS4050 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	Fort Dearborn operates software tools as part of a process by which Fort Dearborn generates, references, and/or incorporates VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below. Each of these files further defines at least one variable data area, as described further below. Examples of software used to generate VDP files include GMC Printnet, and the HP SmartStream Designer for Adobe InDesign or Quark Xpress.

'233 Patent, Claim 14	
	In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by Fort Dearborn. Refs. [3]-[9]. The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets.

'233 Patent, Claim 14	
	Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for

'233 Patent, Claim 14	
	instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) providing a merge file including a plurality of variable data items;	The VDP files can use variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which stores data either internally or externally. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. Ref. [17] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) processing the page description language file, and during the processing step, generating a static bitmap of the static data area and associating the variable data area with the plurality of variable data items; and	Fort Dearborn runs software on dedicated print servers or digital front ends to parse the VDP files that it generates and/or receives. Each of the HP digital presses operated by Fort Dearborn includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. Fort Dearborn uses such dedicated print servers or digital front ends to process VDP files

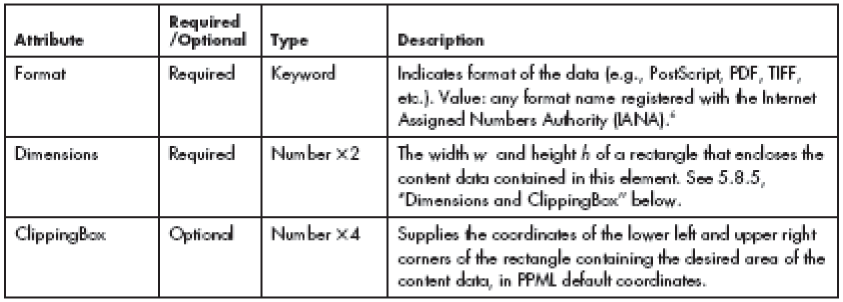
'233 Patent, Claim 14	
	including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point

'233 Patent, Claim 14	
	to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to associate variable data areas with variable data items. For example, in PDF/VT files, document part metadata provides field name information for variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and

'233 Patent, Claim 14	
	EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data. Ref. [12] at 42-43. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.
(d) saving the static bitmap;	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the

'233 Patent, Claim 14	
	system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) generating a first variable data bitmap of a first one of the variable data items utilizing a graphics state associated with the variable data area;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively

'233 Patent, Claim 14	
	define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'233 Patent, Claim 14</th><th></th></tr>
<tr><td></td><td>
<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p></td></tr>
<tr><td>(f) merging the first variable data bitmap with a copy of the static bitmap to produce a first output bitmap;</td><td><p>Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. <i>See</i> Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap.</p>
<p>For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient.</p></td></tr>
</table>

'233 Patent, Claim 14	
	Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a next variable data bitmap of a next one of the variable data items utilizing a graphics state associated with the variable data area;	Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document, according to the contentions with respect to element (e). Appearance information is reused for each instance of the document. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. Fort Dearborn runs software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data

'233 Patent, Claim 14	
	bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. Fort Dearborn may use other VDP file types with infringing characteristics, features, and functions

'233 Patent, Claim 14	
	similar to those described above in these exemplary file types.
and (h) merging the next variable data bitmap with a copy of the static bitmap to produce a next output bitmap;	The print server or digital front end merges the variable data bitmaps with the template bitmap according to the contentions with respect to element (f). The appearance information and the template bitmap are reused for each instance of the document. The template bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. The template bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins.

'233 Patent, Claim 14	
	PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2.
and (i) repeating steps (g) (h) for remaining variable data items in the plurality of variable data items.	The activities performed for steps (g) and (h) are repeated for each remaining variable data item in the plurality of data items.

References:

The following references provide exemplary support for IPT's infringement contentions and are cited throughout the charts below. Support provided within the specific pages and/or paragraphs cited below is not to be interpreted in any way to limit IPT's infringement contentions. IPT reserves the right to support its infringement contentions with information provided in any of the documents listed below. These contentions are based solely on publicly available information relating to exemplary variable data ("VDP") files and raster image processor (RIP") software and press control software used on HP's presses. Discovery in this case has not yet begun, and the charts below do not reflect any information produced by defendantsFort Dearborn Company and Hewlett-Packard Company. Other VDP files, RIP software and press control software may be identified through discovery as being used by defendants. IPT reserves the right to support its contentions with additional material produced by the defendants or subsequently identified by IPT.

[1] HP Indigo Production Manager: Flexible Scalable Digital Front End For High Volume, Complex Jobs, available at http://h10088.www1.hp.com/gap/en/4AA1-0277ENUS_Production%20Mngr_Low%20Res_Feb%202007.pdf
[2] HP SmartStream, available at http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-9528EEW.pdf
[3] HP T200 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T200_Data_Sheet.pdf
[4] HP T300 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T300_Data_Sheet.pdf
[5] HP T350 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T350_Data_Sheet.pdf
[6] HP T400 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2011/HPInkjetPremiere/T400_Data_Sheet.pdf
[7] HP Indigo w3250 Data Sheet, available at http://h10010.www1.hp.com/wwpc/pscmisc/vac/us/product_pdfs/90566.pdf
[8] HP Indigo 5600 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2012/HPPredrupa12/HP_Indigo_5600.pdf
[9] HP Indigo 7500 Data Sheet, available at http://www.hp.com/hpinfo/newsroom/press_kits/2010/IPEX2010/HP_Indigo_7500_DS.PDF
[10] PPML Template, available at: www.standards.podi.org/component/docman/doc_download/8-ppmltemplate-v110-2002-12-12.html
[11] PPML Specification v1.5, available at http://www.standards.podi.org/ppml/specification.html
[12] PPML Specification v2.1, available at http://www.standards.podi.org/ppml/specification.html

[13] Global Graphics/Harlequin White Paper "High Performance Variable Data Printing using PDF" http://www.globalgraphics.com/pdf/products/variable-data-printing-using-pdf.pdf
[14] HP Indigo Yours Truly Designer 7 User Guide
[15] Harper, Elliott, "Speaking in Tongues: Sorting Out Variable Data Printing Languages" THE SEYBOLD REPORT, Vol. 7, No. 17 (Sep. 6, 2007), available at http://www.fujixerox.com.au/products/image/media/TSR-0906-Speak-Tongues-reprint.pdf.
[16] Adobe Systems Inc., PDF Reference 5th Ed., v. 1.6, available at http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference16.pdf
[17] ISO 16612-2:2010, available at http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=46428
[18] Global Graphics, Do PDF/VT Right, available at http://www.globalgraphics.com/doPDFVTright/

U.S. Patent No. 5,729,665 ("the '665 patent")

'665 Patent, Claim 1	
1. A method for generating multiple bit maps suitable for high-speed printing or plate-making comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '665 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the '665 patent and of the technology claimed within the '665 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '665 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or

‘665 Patent, Claim 1	
	encourage direct infringement of the ‘665 patent. HP, Fort Dearborn, and HP’s other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services and resulting printed products to their customers. HP, Fort Dearborn, and HP’s other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP’s other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor (“RIP”) software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP’s other customers operate digital presses manufactured by HP, including without limitation HP’s Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code representing a template, said page description code defining at least one variable data area and said page description code further defining a graphics state corresponding to said variable data area, said graphics state including at least one attribute which controls the appearance of variable data in said variable data area;	HP and Fort Dearborn and other HP customers operate software tools as part of a process by which HP, Fort Dearborn, and HP’s other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in element (b) below. Each of these files further defines at least one variable data area, as described further in element (b) below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP’s digital presses such as the ones operated by HP and Fort Dearborn and other HP customers. Refs. [3]-[9].

'665 Patent, Claim 1	
	To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn and HP's other customers enter contracts with these third parties through which Fort Dearborn and HP's other customers enforce the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed

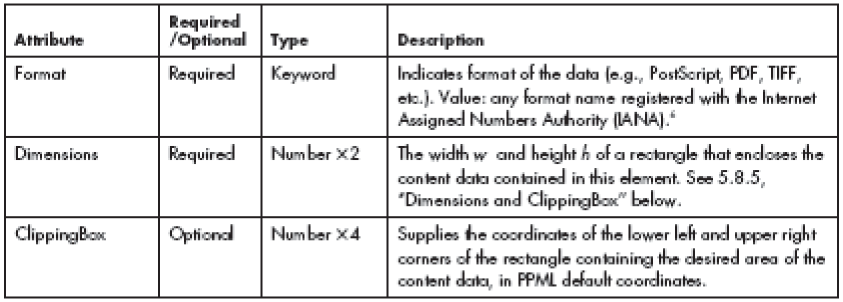
‘665 Patent, Claim 1	
	on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-

‘665 Patent, Claim 1	
	spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) executing said page description code to generate a bit map of said template, and during said execution, identifying said variable data area defined by said page description code and reserving said graphics state corresponding to said variable data area upon said identification;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor (“RIP”) software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the

‘665 Patent, Claim 1	
	document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and

‘665 Patent, Claim 1	
	treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag “associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to “content packages” that “include any static content in the file (text and image page objects, for instance).” Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT

'665 Patent, Claim 1	
	include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) retrieving variable data;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

‘665 Patent, Claim 1	
	types.
(d) associating said variable data with said graphics state corresponding to said variable data area;	HP, Fort Dearborn, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to associate the appearance information found in the VDP file to the corresponding variable data. As described above, variable data may be stored within the VDP file or in one or more separate files. The RIP software associates the variable data with the appearance information defined for the variable data area, as described further in element (e) below.
(e) applying said graphics state corresponding to said variable data area to said variable data to generate a variable data bit map; and	HP, Fort Dearborn, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP’s print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an

‘665 Patent, Claim 1

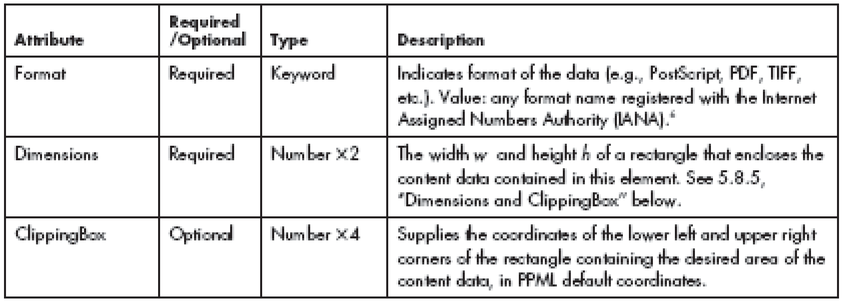
Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34.

“The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The

'665 Patent, Claim 1	
	appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46. In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) merging said variable data bit map with said bit map of said template;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10.

'665 Patent, Claim 1	
	In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
wherein said graphics state corresponding to said variable data area is applied repeatedly to variable data to generate a multitude of variable data bit maps without the need to repeat said executing step (b).	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the

‘665 Patent, Claim 1	
	document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT “for-each” command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

‘665 Patent, Claim 12	
12. The method of claim 1 wherein said reserving, retrieving, associated, applying, and merging steps are repeated for each variable data area defined by said page description code.	VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end repeats the steps recited in claim 1 for each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. “Do” statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics

'665 Patent, Claim 12	
	state and template information on each additional page. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving, retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'665 Patent, Claim 13	
13. The method of claim 12 wherein said reserving, retrieving, associating, applying and merging steps are activated by a control task running in a printer controller, and wherein said control task interrupts said page description code execution	As described above, the steps of reserving, retrieving, associating, applying and merging are all activated and monitored by a control task running in a dedicated print server or digital front end associated with the press. Each of the respective print servers or digital front ends runs RIP software such as the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The control task interrupts said page description code execution upon identifying a predetermined command in said page description code to enable other operations to be performed.

‘665 Patent, Claim 13	
upon identifying a predetermined command in said page description code.	

‘665 Patent, Claim 20	
20. A method for generating a plurality of bit maps suitable for high-speed printing or plate-making from a page description code representing a template and defining at least one variable data area, and from a merge file containing a plurality of data records of at least one variable data field type, the method comprising the steps of:	Defendant Hewlett-Packard (“HP”), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print (“VDP”) jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn’s facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP’s Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the ‘665 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the ‘665 patent and of the technology claimed within the ‘665 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the ‘665 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the ‘665 patent. HP, Fort Dearborn, and HP’s other customers directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services and resulting printed products to their customers. HP, Fort Dearborn, and HP’s other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort

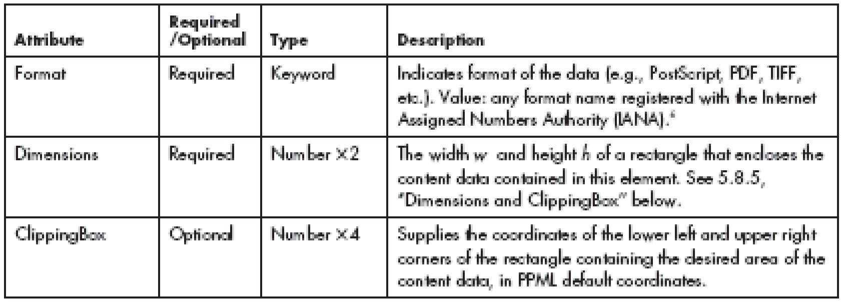
'665 Patent, Claim 20	
	Dearborn, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP's other customers operate digital presses manufactured by HP, , including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the "executing a control task" step below. Each of these files further defines at least one variable data area, as described further in the "executing a control task" step below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9]. To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn and HP's other customers enter contracts with these third parties through which Fort Dearborn and HP's other customers enforce the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation,

'665 Patent, Claim 20	
	font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.

‘665 Patent, Claim 20

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

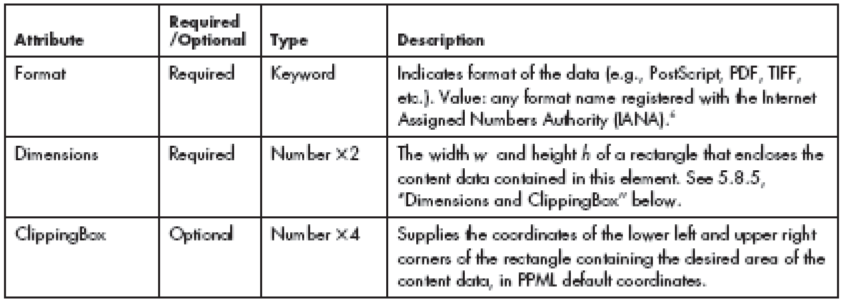
In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

‘665 Patent, Claim 20	
	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be referenced from outside of the PPMLT file itself. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., “<xsl: value-of select='name'/>” points to a database entry for the “name” element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a page description code interpretive program, said interpretive program generates graphics states for each data area defined by said page	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production

'665 Patent, Claim 20	
description code;	Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The RIP software necessarily includes a module or other discrete software component that interprets VDP files, including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The

‘665 Patent, Claim 20	
	format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39. “The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'665 Patent, Claim 20</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width <i>w</i> and height <i>h</i> of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>executing a control task in conjunction with said interpretive program, said control task identifies said variable data area defined by said page description code and</td><td>As described above, HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends. RIP software identifies variable data areas defined in VDP files. The RIP software necessarily includes one or more module or other discrete software component that identifies variable data areas, reserves graphics states, and generates one or more template bitmap, and saves one or more template bitmap in memory.</td></tr>
</table>

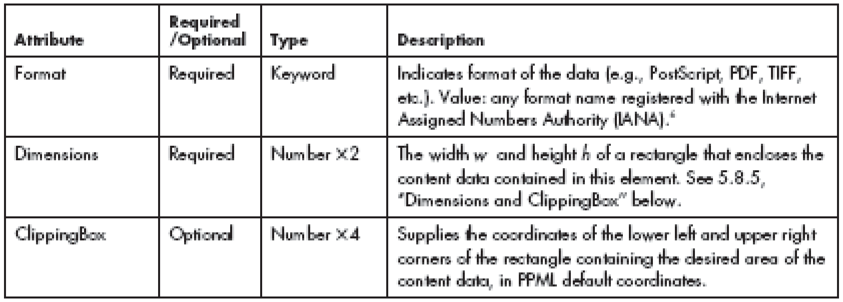
'665 Patent, Claim 20	
reserves said graphics states generated by said interpretive program for said variable data area, said control task generates a template bit map defined by said page description code, and after the completion of said interpretive program, said control task saves said template bit map in memory; and	HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a

'665 Patent, Claim 20	
	document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some

'665 Patent, Claim 20	
	or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
executing a merge task upon completion of said interpretive program, said merge task generates variable data bit maps for said data records in said merge file by applying said reserved graphics states to said data records, and said merge task merges said variable data bit maps with a separate copy of said template bit map to create the plurality of bit maps suitable for high-speed printing or plate making;	As described above, HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends. RIP software applies appearance information found in the VDP file to the corresponding variable data areas. The RIP software necessarily includes one or more module or other discrete software component that applies the appearance information to the corresponding variable data to generate a variable data bit map. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g.,

'665 Patent, Claim 20	
	PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

‘665 Patent, Claim 20



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include “channels”, which define the position, scaling, and rotation of separately defined “content packages.” Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Fort Dearborn, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap.

For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5.

In another example, “PPML constructs a page image by placing a series of Marks on the page.

‘665 Patent, Claim 20	
	Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction.” Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define “channels” that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. HP, Fort Dearborn, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
whereby said reserved graphics states are applied repeatedly to said data records to generate said variable data bit maps for said data records without the need to repeat said steps of executing a page description code interpretive program and executing a control task in conjunction with said interpretive program.	HP, Fort Dearborn, and HP’s other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. “Do” statements refer back to XObjects that define objects that are used repeatedly,

‘665 Patent, Claim 20	
	allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT “for-each” command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4.

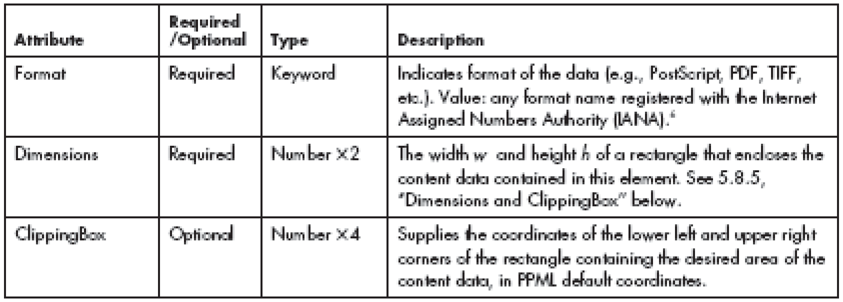
U.S. Patent No. 5,937,153 ("the '153 patent")

'153 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '153 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the '153 patent and of the technology claimed within the '153 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '153 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '153 patent. HP, Fort Dearborn, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Fort Dearborn, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP's other customers operate digital presses

'153 Patent, Claim 1	
	manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) generating a page description code specification, the page description code specification defining at least one data area to become variable, and the page description code further defining a graphics state corresponding to the data area, the graphics state including at least one attribute which controls the appearance of data in the data area;	HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in element (b) below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9]. To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn and HP's other customers enter contracts with these third parties through which Fort Dearborn and HP's other customers enforce the obligations that it imposes upon third-parties. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states).

'153 Patent, Claim 1	
	Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA,

<table>
<tr><th>'153 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>
INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

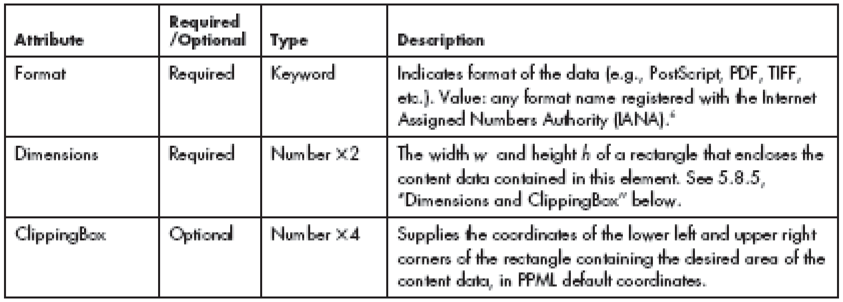
HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(b) interpreting the page</td><td>HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital</td></tr>
</table>

'153 Patent, Claim 1	
description code specification, and during the interpretation, identifying the data area defined by the page description code specification;	front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for

'153 Patent, Claim 1	
	instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) storing the graphics state corresponding to the data area upon the identification of the variable data area in step (b);	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) retrieving a variable data item from a plurality of variable data items;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable

'153 Patent, Claim 1	
	OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) applying the stored graphics state to the variable data item to generate a variable data bit map; and	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that

'153 Patent, Claim 1	
	encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'153 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>
<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p></td></tr>
<tr><td>(f) repeating steps (d) and (e) for remaining variable data items in the plurality of variable data items, whereby the stored graphics state is applied</td><td>HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document.</td></tr>
</table>

'153 Patent, Claim 1	
repeatedly to generate a plurality of variable data bit maps.	The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of

'153 Patent, Claim 1	
	variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 3	
3. The computer implemented method of claim 1, wherein the page description code specification represents a template and includes a static data area; and the computer implemented method further comprises the steps of:	As described for claim 1 of the '153 patent, HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files. Each of these files represents a template. These VDP files use static data areas to quickly manage VDP jobs. PPML for example, performs more efficiently when the static data areas are defined in advance. Ref. [10] at 10.
executing portions of the page description code specification corresponding to the static data area to generate a template bit map;	HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP

'153 Patent, Claim 3	
	software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
storing the template bit map; and	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by

'153 Patent, Claim 3	
	avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing

'153 Patent, Claim 3	
	characteristics, features, and functions similar to those described above in these exemplary file types.
merging each of the plurality of the variable data bit maps into a clean copy of the template bit map to create a plurality of merged bit maps.	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 4	
4. The computer implemented method of claim 1, wherein the	As described for claim 1 of the '153 patent, the controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type

'153 Patent, Claim 4	
identifying step includes the step of detecting predefined characters within a text string defined in the page description code specification.	of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files use objects denoted by the text "obj" to identify template and variable data areas. Further, the text "/XObject" denotes information in certain objects that will be reused. The RIP software may detect these characters or the RIP software may evaluate repetitive text within the PDF files to identify data areas. In PDF/VT, XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." [17] at § 6.7.3. For example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [12] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [12] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [12] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [17] at 4-5. JLYT files include channels that define links to variable content. Ref. [17] at 5. Each of these structures is associated with a predetermined characters within the JLYT file. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 5	
5. The computer implemented	As described above, PDF, PDF/VT, PPML, PPMLT, and JLYT can each define appearance

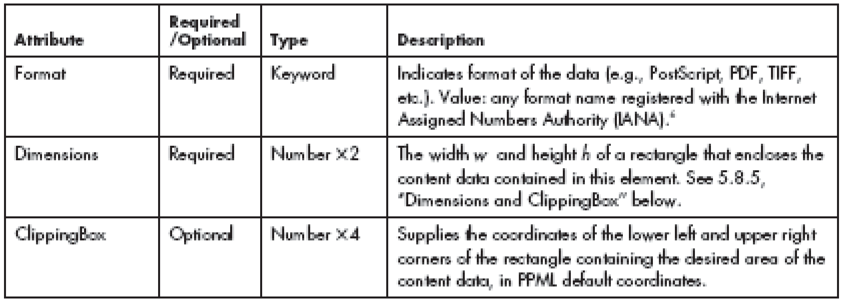
'153 Patent, Claim 5	
method of claim 1, wherein the attribute is a size attribute, a font attribute, a position attribute, an orientation attribute or a location attribute.	information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation

'153 Patent, Claim 5	

matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37.

"The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39.

"The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text

'153 Patent, Claim 5	
	or variable text fields. See Ref. [14] at 16. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'153 Patent, Claim 6	
6. A computer implemented method for processing a page description code specification comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '153 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the '153 patent and of the technology claimed within the '153 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '153 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '153 patent. HP, Fort Dearborn, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Fort Dearborn, and HP's other customers operate software capable of generating, referencing, and/or

'153 Patent, Claim 6	
	incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP's other customers operate digital presses manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See*, *e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template, as described further below. Each of these files further defines at least one variable data area, as described further in the "interpreting" step below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9]. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store

'153 Patent, Claim 6	
	the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.
interpreting the page description code specification, and during	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses

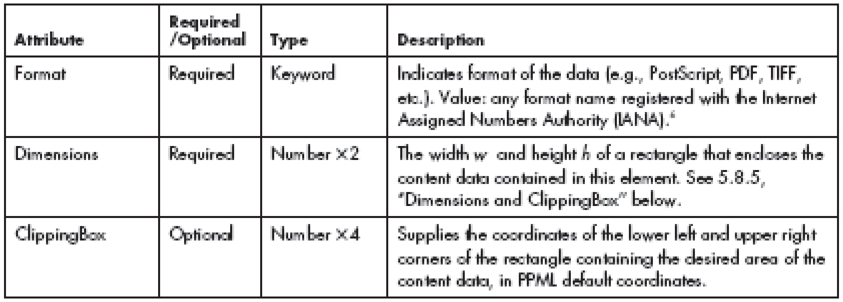
'153 Patent, Claim 6	
the interpretation, identifying a data area defined by the page description code specification;	operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to

'153 Patent, Claim 6	
	identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
upon the identification of the data area, storing a graphics state set forth in the page description code specification which defines an attribute of how data is to appear in the data area; and	The VDP file also defines information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. The appearance information remains unchanged from document to document regardless of whether the corresponding text changes. Since the appearance information is static, it is stored and used repeatedly to render the associated variable data. VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects, include the capability of defining appearance information such that it can be reused. For example, PDF and PDF/VT define stored dictionary resources including graphics state parameters, as described above. [16] at § 4.3.4. Likewise, PPML and PPMLT include the SUPPLIED_RESOURCE and SUPPLIED_RESOURC_REF elements, which allow definition of fonts for later reuse. [11] at 105-106; [12] at 113-114. As a further example, JLYT files define stored channels that include scaling and rotation parameters for each element. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
repeatedly retrieving data records from a plurality of data records and applying the stored	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends

'153 Patent, Claim 6	
graphics state to the data records to generate a plurality of bitmaps of the data records so that the bitmaps of the data records include the attribute.	running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font,

'153 Patent, Claim 6	
	font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

‘153 Patent, Claim 6	



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document.

The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused

'153 Patent, Claim 6	
	for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

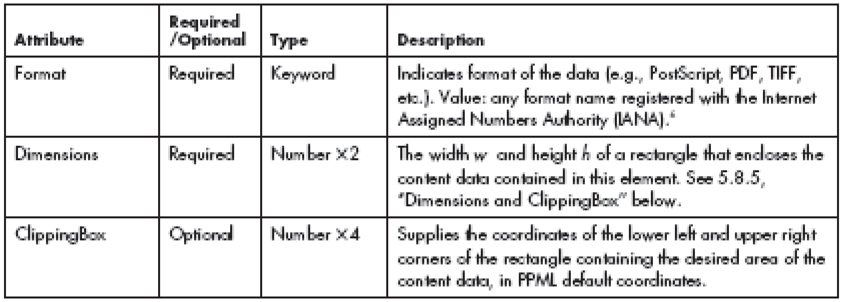
‘153 Patent, Claim 6	
	types.

U.S. Patent No. 6,381,028 ("the '028 patent")

'028 Patent, Claim 1	
1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '028 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the '028 patent and of the technology claimed within the '028 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '028 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '028 patent. HP, Fort Dearborn, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Fort Dearborn, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP's other customers operate digital presses

'028 Patent, Claim 1	
	manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description code specification, the page description code specification defining at least one data area, and the page description code further defining a graphics state including at least one attribute which controls the appearance of data in the data area;	HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these files defines at least one variable data area, as described further in step (b) below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet and Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9]. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which

'028 Patent, Claim 1	
	contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'028 Patent, Claim 1</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).*</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>
Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(b) interpreting the page description code specification, and during the interpretation step, identifying the data area defined by the page description</td><td>HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production</td></tr>
</table>

'028 Patent, Claim 1	
code specification;	Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

'028 Patent, Claim 1	
	types.
(c) upon the identification of the data area in step (b), applying the graphics state corresponding to the data area to a set of alphanumeric characters so as to generate a plurality of character bit maps;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to characters associated with the variable data areas. The appearance information is applied to the characters by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See*, *e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT,

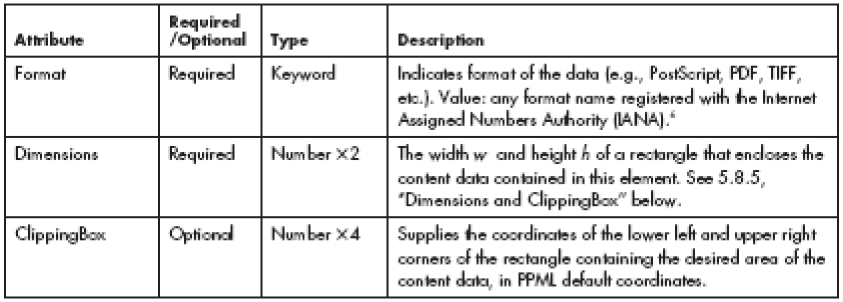
‘028 Patent, Claim 1	

REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36.

“The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37.

“The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39.

“The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.



Attribute	Required /Optional	Type	Description
Format	Required	Keyword	Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]
Dimensions	Required	Number ×2	The width *w* and height *h* of a rectangle that encloses the content data contained in this element. See 5.8.5, “Dimensions and ClippingBox” below.
ClippingBox	Optional	Number ×4	Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press’s proprietary format, and allows for the full use of HP Indigo Press features and

'028 Patent, Claim 1	
	optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16. RIP software applies the graphics state as part of generating character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(d) storing the plurality of character bit maps;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to store character bitmaps. The character bitmaps are stored by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software stores the character bitmaps for each character that appears within a given font associated with a variable data area. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for

'028 Patent, Claim 1	
	every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) retrieving a variable data item from a plurality of variable data items;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4.

'028 Patent, Claim 1	
	HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) associating the variable data item with the plurality of character bit maps;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to associate variable data items with the character bitmaps. The variable data items associated with character bitmaps are identified by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. RIP software necessarily associates the character bitmaps for each character in the respective variable data areas. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a variable data bit map for the variable data using the character bit maps; and	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to generate variable data bitmaps. The variable data bitmaps are generated by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers.

'028 Patent, Claim 1	
	RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(h) repeating steps (e) through (g) for remaining variable data items in the plurality of variable data items, whereby the stored character bit maps are used repeatedly to generate a plurality of variable data bit maps.	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to use the stored character bitmaps for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the stored character bitmaps and the template bitmap are reused for each instance of the document. As discussed above, RIP software uses and reuses the character bitmaps to reduce the processing that must be done when rendering variable data bitmaps, as explained in the references. For example, PDF and PDF/VT files are designed such that "efficient implementation can be achieved through careful caching and reuse of previously rendered glyphs." Ref. [16] at 358. In a whitepaper describing best practices for PDF/VT, Global Graphics explains that fonts are preferably the same for each variable data area. In instances where different fonts are assigned, "the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly." Ref. [18] at 58. In the example of PPML or PPMLT files that reference PDF files, the RIP software would incorporate the same approach as described above. As another example, PPML, PPMLT, and JLYT files are likely to cache character bitmaps to avoid the burden of re-rendering the characters for each variable data area.

'028 Patent, Claim 1	
	HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'028 Patent, Claim 2	
2. The computer implemented method of claim 1, wherein the page description code specification represents a template and includes a static data area, and the computer implemented method further comprises the steps of:	The elements of claim 1 are described in the chart above. HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files represents a template and includes a static data area, as described further in the "executing" step below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are among the file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9].
executing portions of the page description code specification corresponding to the static data area to generate a template bit map; and	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends

'028 Patent, Claim 2	
	to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point

'028 Patent, Claim 2	
	to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.
merging each of the plurality of the variable data bit maps into clean copies of the template bit map to create a plurality of merged bit maps.	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5.

'028 Patent, Claim 4	
4. A computer implemented method for generating a reusable template bit map suitable for high-speed variable printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000,

'028 Patent, Claim 4	
	20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '028 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the '028 patent and of the technology claimed within the '028 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '028 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '028 patent. HP, Fort Dearborn, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Fort Dearborn, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP's other customers operate digital presses manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below. HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are

'028 Patent, Claim 4	
	substantially similar in relevant respects. Each of these VDP files represents a template, as described further in the "executing" step below.
generating a page description code specification, the page description code specification defining at least one variable data area and at least one static data area;	HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further in the "executing" step below. Each of these files further defines at least one variable data area, as described further in the "identifying" step below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet, and the HP SmartSTream Designer for Adobe InDesign or Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9]. To the extent that third-parties, such as Fort Dearborn's customers and/or their print media agents, perform the step of generating these files, Fort Dearborn and HP's other customers direct and control such third-parties, for example, by dictating the manner by which the third-parties must supply data to enable VDP jobs. Further, upon information and belief, Fort Dearborn and HP's other customers enter contracts with these third parties through which Fort Dearborn and HP's other customers enforce the obligations that it imposes upon third-parties. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description code specification, and during the interpreting step,	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production

'028 Patent, Claim 4	
	Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers.
generating a bitmap of the static data area and adding the bitmap of the static data area to a template bitmap;	HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped

‘028 Patent, Claim 4	
	(prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing

'028 Patent, Claim 4	
	characteristics, features, and functions similar to those described above in these exemplary file types.
identifying the variable data area, and	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

'028 Patent, Claim 4	
	types.
responsive to the identification of the variable data, not adding a bitmap of the variable data area to the template bitmap; and	As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets, and therefore does not include a bitmap of the variable data area. Adding a bitmap of the variable data area to the template bitmap would prevent reuse of the static bitmap. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
saving the template bitmap, whereby copies of the template bitmap can be continuously accessed to create a plurality of variable data bitmaps.	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins.

'028 Patent, Claim 4	
	PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

U.S. Patent No. 7,274,479 ("the '479 patent")

'479 Patent, Claim 9	
9. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '479 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the '479 patent and of the technology claimed within the '479 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '479 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '479 patent. HP, Fort Dearborn, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Fort Dearborn, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP's other customers operate digital presses

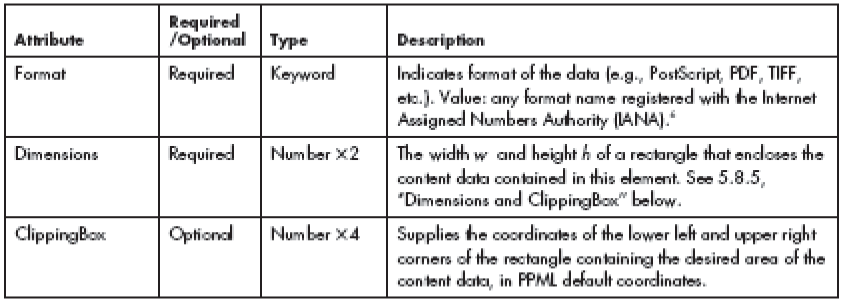
'479 Patent, Claim 9	
	manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a print specification, the print specification defining at least one variable data area and at least one static data area;	HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a static data area, as described further below. Each of these files further defines at least one variable data area, as described further in element (b) below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9]. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet

‘479 Patent, Claim 9	
	presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(b) providing a plurality of variable data items;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by

'479 Patent, Claim 9	
	Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) identifying the variable data area;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo,

'479 Patent, Claim 9	
	or Esko installed on HP's print servers or digital front end computers. HP, Fort Dearborn, and HP's other customers use such print servers or digital front ends to process VDP files including one or more of PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data, as described above in element (a). The VDP file defines variable data areas based on the surrounding tags of the data element.
(d) associating a graphic state with the variable data area, the graphic state including at least one attribute controlling the appearance of items to be printed in the variable data area;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to associate appearance information found in the VDP file to the corresponding variable data. The VDP file includes information such as the size and location for each variable data element and includes graphics state information including appearance information such as spacing, rotation, font, word spacing, letter spacing, justification, and color for variable data. Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types, for example, are capable of encoding some or all of these appearance attributes. Each of the VDP files defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data. For example, PDF and PDF/VT include graphics state operators and text state operators that define appearance information of graphics and text within variable data areas defined in PDF or PDF/VT files. [16] at 180-194 (describing the graphics state), 366-373 (describing text states). Appearance of every graphics object, including text, defined by a PDF or PDF/VT file is controlled by the graphics state, which defines color (color parameter); position, rotation, and skew (via a transformation matrix); line characteristics including line width and dash patterns; text font (Tf parameter), text font size (Tfs parameter), word spacing (Tw parameter), and character spacing (Tc parameter). In another example, PPML files include elements that define one or more jobs, each of which contains one or more documents. Each document contains one or more pages, and each page includes one or more objects that represent reusable data areas or non-reusable data areas. The MARK element and the elements it encloses collectively define the appearance of the object to be

'479 Patent, Claim 9	
	printed. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered. . . . VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>


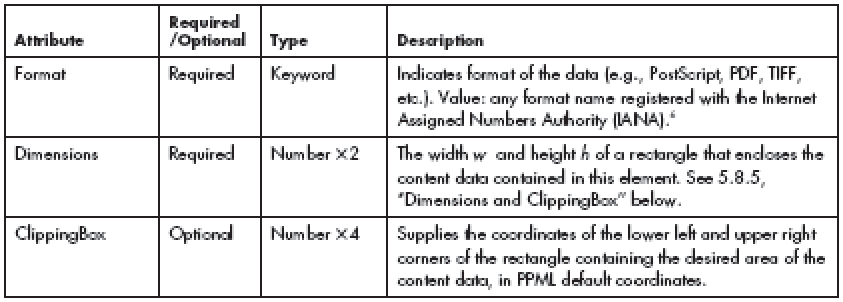
<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

<p>Ref. [11] at 28; Ref. [12] at 40.</p>
<p>In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.</p>
<p>In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.</p>
<p>As described above in element (b), variable data may be stored within the VDP file or in one or more separate files. Each field retrieved from a variable data record is matched to the corresponding variable data area defined within the VDP file. For example, "Name" data in a given record is matched to variable data areas that are associated in the file with the "Name" field.</p>
<p>HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.</p>
</td></tr>
<tr><td>(e) retrieving a variable data</td><td>HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital</td></tr>
</table>

'479 Patent, Claim 9	
item from the plurality of variable data items;	front ends, as described above, to retrieve variable data elements stored within the VDP file or in one or more separate files. The variable data is retrieved by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. For example, PDF and PDF/VT files define variable data within the file itself or by reference to external resources. In PDF and PDF/VT files, the RIP software retrieves objects and XObjects that are not repeated. Further, in PDF/VT files, DPart nodes with variable data are retrieved by the RIP software. In another example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which is retrieved by the print servers or digital front ends. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print servers or digital front ends. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(f) generating a bitmap for the variable item, the generating step including a step of applying the graphic state associated with the variable data area to the variable data item; and	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at

'479 Patent, Claim 9	
	2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: "The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones." Ref. [11] at 22; Ref. [12] at 34. "The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE." Ref. [11] at 24; Ref. [12] at 36. "The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW." Ref. [11] at 25; Ref. [12] at 37. "The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27; Ref. [12] at 39. "The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data.

<table>
<tr><th>'479 Patent, Claim 9</th><th></th></tr>
<tr><td></td><td>
The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object." Ref. [11] at 28; Ref. [12] at 40.



<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[6]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width w and height h of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(g) repeating steps (e) and (f) for remaining variable data</td><td>HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to</td></tr>
</table>

'479 Patent, Claim 9	
items in the plurality of variable data items, whereby the graphic state associated with the variable data area is applied repeatedly to generate a plurality of variable data bitmaps.	the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54.

'479 Patent, Claim 9	
	In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 10	
10. The method of claim 9, wherein the graphic state associated with the variable data area is defined within the print specification.	Each of the PDF, PDF/VT, PPML, PPMLT, and JLYT file types defines appearance information such as spacing, size, location, rotation, font, word spacing, letter spacing, justification, and color for static and variable data, as discussed above with respect to element (d) of claim 9 of the '479 Patent. The appearance information may be defined within the print specification either by referencing an external file or by providing the appearance information directly within the VDP file.

'479 Patent, Claim 15	
15. The method of claim 9, wherein the variable data area and the static data area are defined, at least in part, by page description language commands.	As described for claim 9 of the '479 Patent, the VDP file defines static and variable data areas based on the surrounding tags of the data element. VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects each incorporate page description language commands. Each of these files is a page description language file, and the tags and commands included in each of these files are therefore page description language commands. The VDP file defines static and variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing, as described in elements (a) and (b) of claim 9.

'479 Patent, Claim 15	
	HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 17	
17. The method of claim 9, further comprising a step of caching a representation of the static data area, whereby the cached representation of the static data area is available for merging with the variable data bitmaps to generate merged documents.	A static bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and

'479 Patent, Claim 17	
	can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5.

'479 Patent, Claim 18	
18. The method of claim 17, wherein the cached representation of the static data area is a bitmap representation.	The cached representation of the static data area is a bitmap to avoid the redundant burden of the system to continually compute the contents appearance, as discussed above for claim 17 of the '479 Patent. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents,

'479 Patent, Claim 18	
	Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

'479 Patent, Claim 19	
19. The method of claim 9, wherein: the plurality of data items are associated with a field name; and	As described for claim 9 of the '479 Patent, each field retrieved from a variable data record is matched to a corresponding named variable data area defined within the VDP file.
the step of identifying a variable data area includes the step of detecting, in the print specification, a character string associated with the variable data	The controller identifies variable data elements by scanning the variable data files and finding the tags associated with such variable data. The type of tag depends upon the type of VDP file that the controller is processing. For example, in PDF/VT files, document part metadata provides field name information for

'479 Patent, Claim 19	
area that matches the field name associated with the plurality of data items.	variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data. Ref. [12] at 42-43. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.

U.S. Patent No. 7,333,233 ("the '233 patent")

'233 Patent, Claim 12	
12. A computer implemented method for generating a static bitmap suitable for high-speed variable printing, comprising the steps of:	Defendant Hewlett-Packard ("HP"), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print ("VDP") jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn's facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the '233 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the '233 patent and of the technology claimed within the '233 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the '233 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the '233 patent. HP, Fort Dearborn, and HP's other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing ("VDP") services and resulting printed products to their customers. HP, Fort Dearborn, and HP's other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP's other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor ("RIP") software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP's other customers operate digital presses

‘233 Patent, Claim 12	
	manufactured by HP, including without limitation HP's Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See*, *e.g*, Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	HP, Fort Dearborn, and HP's other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP's other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below, and in the "interpreting" step. Each of these files further defines at least one variable data area, as described further below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP's digital presses such as the ones operated by HP, Fort Dearborn, and HP's other customers. Refs. [3]-[9]. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex

'233 Patent, Claim 12	
	C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. The VDP file defines static data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a

'233 Patent, Claim 12	
	REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
interpreting the page description language file, and during the interpreting step, generating a static bitmap of the static data area;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo,

'233 Patent, Claim 12	
	or Esko installed on HP's print servers or digital front end computers. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template bitmap. The template bitmap is composed of reusable elements within a given job. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. The VDP file defines static data areas based on the surrounding tags of the data element. The type

'233 Patent, Claim 12	
	of tag depends upon the type of VDP file that the controller is processing. For example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
and saving the static bitmap, whereby the saved static bitmap is used repeatedly in the generation of a plurality of documents, each of which contains the static bitmap and a variable data bitmap.	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. *See* Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap. For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these

'233 Patent, Claim 12	
	items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it

'233 Patent, Claim 12	
	takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. VDP files are optimized for handling variable data associated with a series of documents. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as an example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these are necessarily processed according to the reserving, retrieving, associated, and applying steps before being merged with the one or more static bitmaps of the template. Ref. [11] at 15. PPMLT is structured and processed similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data

‘233 Patent, Claim 12	
	area present in a PPMLT file, an embedded XSLT “for-each” command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap template. Ref. [15] at 4. HP, Fort Dearborn, and HP’s other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.

‘233 Patent, Claim 14	
14. A computer implemented method for generating a plurality of bitmaps suitable for high-speed printing, comprising the steps of:	Defendant Hewlett-Packard (“HP”), directly and/or through its subsidiaries, affiliates, agents, and/or business partners, has in the past and continues to directly infringe by setting up and running variable data print (“VDP”) jobs including at tradeshows, tech centers, sales centers, product demonstrations, open houses and at Fort Dearborn’s facilities, including by operating at least Indigo Digital Presses supplied by HP, including: HP’s Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. HP also induces Fort Dearborn and other HP customers to commit direct infringement by one or more of supplying, offering for sale and selling its Inkjet Web Presses, and its Indigo Digital Presses. Each of these presses was designed and intended to practice methods covered by the ‘233 patent, and, on information and belief, HP has supplied related training and support materials and services to Fort Dearborn and other HP customers. Despite its awareness of the ‘233 patent and of the technology claimed within the ‘233 patent, HP has continued these acts of inducement with specific intent to cause and/or encourage such direct infringement of the ‘233 patent and/or with deliberate indifference of a known risk or willful blindness that such activities would cause and/or encourage direct infringement of the ‘233 patent.

‘233 Patent, Claim 14	
	HP, Fort Dearborn, and HP’s other customers, directly and/or through their subsidiaries, affiliates, agents, and/or business partners, have in the past and continue to directly infringe by setting up and running variable data print jobs and by selling and/or offering to sell related variable data printing (“VDP”) services and resulting printed products to their customers. HP, Fort Dearborn, and HP’s other customers operate software capable of generating, referencing, and/or incorporating VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. In addition to software, HP, Fort Dearborn, and HP’s other customers operate presses with dedicated print servers or digital front ends that process VDP jobs using raster image processor (“RIP”) software provided by HP or a third-party. For example, HP, Fort Dearborn, and HP’s other customers operate digital presses manufactured by HP, including without limitation HP’s Inkjet Web Presses, e.g., T200, T300, T350 and T400 presses and its Indigo Digital Presses, e.g., W3050, W3250, 3550, WS4000, WS4050, WS4600, 5000, 5600, 7200, WS6600, WS6600p, W7200, W7250, 7500, 7600, 10000, 20000, and 30000 presses. *See, e.g,* Refs. [1]-[9]. Each of these digital presses receives and processes input files at a print server or digital front-end using RIP software, as further described below.
(a) providing a page description language file, the page description language file defining at least one variable data area and at least one static data area;	HP, Fort Dearborn, and HP’s other customers operate software tools as part of a process by which HP, Fort Dearborn, and HP’s other customers generate, reference, and/or incorporate VDP files such as PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects. Each of these VDP files defines a template, as described further below. Each of these files further defines at least one variable data area, as described further below. HP provides at least some software tools that are part of a process by which Fort Dearborn and other HP customers generate, reference, and/or incorporate these VDP files -- including, for example, HP Indigo Yours Truly Designer and HP SmartStream Designer. Other examples of software used to generate VDP files include GMC Printnet Quark Xpress. In addition, PDF, PDF/VT, PPML, PPMLT, and JLYT are file types processed, referenced, and incorporated at a dedicated print server or by a digital front end associated with HP’s digital presses such as the ones operated by HP, Fort Dearborn, and HP’s other customers. Refs. [3]-[9]. The VDP file defines static data areas based on the surrounding tags of the data element. The type

'233 Patent, Claim 14	
	of tag depends upon the type of VDP file that the controller is processing. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and "[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately." Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by "Do" operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is "Record," "File," "Stream," or "Global." Ref. [17] at § 6.7.3. In another example, the OBJECT tag within a PPML file "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. The PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a

'233 Patent, Claim 14	
	document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file

'233 Patent, Claim 14	
	types.
(b) providing a merge file including a plurality of variable data items;	The VDP files can use variable data elements stored internally or in separate files. For example, in PPML documents, variable data is contained within a non-reusable OBJECT tag, which stores data either internally or externally. In another example, in PPMLT documents the DATA tag and DATA_REF tag provides variable data. Ref. [10] at 23-24. Variable data in the PPMLT file may be included internally or externally. Data records and fields internal to the PPMLT file are respectively identified by <R> and <F> tags in PPMLT files. PPMLT files further provide instructions for how to retrieve variable data entries through XSLT scripts embedded in the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. Ref. [17] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(c) processing the page description language file, and during the processing step, generating a static bitmap of the static data area and associating the variable data area with the plurality of variable data items; and	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends to parse the VDP files that they generate and/or receive. Each of the HP digital presses operated by HP, Fort Dearborn, and HP's other customers includes a digital front end capable of executing VDP files. These digital front ends may comprise, for example, an HP SmartStream Onboard Print Server, HP SmartStream Production Pro Print Server, HP SmartStream Production Plus Print Server, HP SmartStream Ultra Print Server, or an HP SmartStream Labels and Packaging Print Server. Each of the respective print servers or digital front ends runs raster image processor ("RIP") software provided by HP or a third-party. The RIP software includes, for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. HP, Fort Dearborn, and HP's other customers use such dedicated print servers or digital front ends to process VDP files including one or more of PDF, PDF/VT, PPML, PPMLT, JLYT files, and/or other VDP file types that are substantially similar in relevant respects; and creates a template

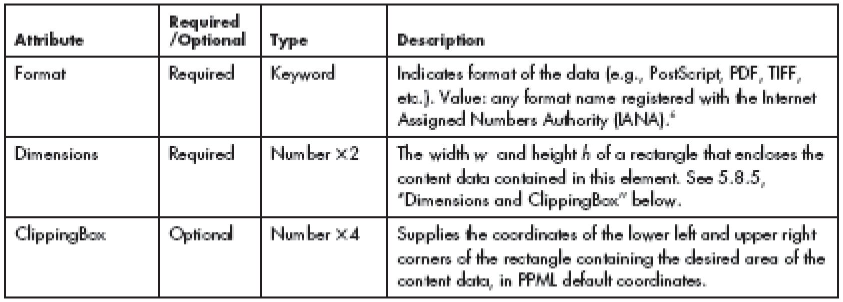
‘233 Patent, Claim 14	
	bitmap. The template bitmap comprises one or more reusable elements defined within the VDP file. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. Ref. [13] at 3, 5. For example, PDF files include information that is repeated for each instance of a document. RIP software provided by HP or third parties is capable of identifying the repeated portions of the document, and optimizing the RIP process by generating a template that includes the repeated portions of the document. For example, the Harlequin RIP software provided with HP inkjet presses identifies shared elements and “[o]nce a shared element has been identified it is only rendered once, while the variable data on each page is rendered separately.” Ref. [13] at 3, 5. In addition to the methods described above for generating a template from a PDF file, PDF/VT files explicitly identify template information by defining XObjects within the PDF/VT file that can be referenced more than once by “Do” operators present in the PDF/VT file. Ref. [17] at § 6.7.1 XObjects may incorporate a GTS_Scope key. Ref. [17] at § 6.7.3. Graphics elements are explicitly identified as reused when the value for the GTS_Scope key is “Record,” “File,” “Stream,” or “Global.” Ref. [17] at § 6.7.3. In another example, the PPML specification explains that “An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML’s productivity improvement.” Ref. [11] at 11; Ref. [12] at 13. “The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content’s appearance.” Ref. [11] at 11; Ref. [12] at 13. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54.

'233 Patent, Claim 14	
	The VDP file defines variable data areas based on the surrounding tags of the data element. The type of tag depends upon the type of VDP file that the controller is processing. For example, PDF and PDF/VT files include objects that define graphics and text areas. By interpreting these objects and the resources or other objects that they refer to, RIP software identifies variable data areas. As discussed above, the RIP software identifies repeated objects and treats them as template data areas. The remaining non-repeated objects are variable data areas. In a further example, PDF/VT files define document part architecture and document part metadata that gives RIP software additional information from which the RIP software identifies variable data areas. Ref. [17] at §§ 6.4, 6.6, Annex C. The document part metadata can identify, for example, the recipient's name, address, ID, and other information. Ref. [17] at §§ 6.4, 6.6, Annex C. In a further example, within a PPML file the OBJECT tag "associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT." Ref. [11] at 27. If the OBJECT tag is contained within a REUSABLE_OBJECT tag, then it denotes a static data area. If the OBJECT tag is contained within a MARK tag then it denotes the start of a variable data area. Ref. [11] at 27 and 33. In yet another example, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. In a further example, JLYT files refer to "content packages" that "include any static content in the file (text and image page objects, for instance)." Ref. [15] at 4-5. JLYT files include channels that define links to variable content. Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to associate variable data areas with variable data items. For example, in PDF/VT files, document part metadata provides field name information for variable data areas. Ref. 17 at § 6.6, Annex C (e.g., CIP4_FirstName, CIP4_LastName, etc.). In another example, within a PPML file the EXTERNAL_DATA and EXTERNAL_DATA_ARRAY elements provide a URI that identifies the source of variable data.

'233 Patent, Claim 14	
	Ref. [12] at 42-43. In yet another example, PPMLT uses TEMPLATE and TEMPLATE_REF elements to identify a document template. Ref. [10] at 20-22. The TEMPLATE and TEMPLATE_REF elements point to a PPML file that has the characteristics explained above. Ref. [10] at 20-22, 41-54. In addition, PPMLT files may include XSL scripting used within OBJECT tags to identify variable data. Ref. [10] at 12-16, 41-54. These XSL scripts may match a variable data item according to a field name encoded within the PPMLT file, e.g., "<xsl: value-of select='name'/>" points to a database entry for the "name" element. Ref. [10] at 27, 37, and 54. In a further example, JLYT files include channels that define links to variable content. Ref. [15] at 5. The links necessarily identify a field name that identifies the plurality of variable data items.
(d) saving the static bitmap;	The static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref.

'233 Patent, Claim 14	
	[11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(e) generating a first variable data bitmap of a first one of the variable data items utilizing a graphics state associated with the variable data area;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply appearance information found in the VDP file to the corresponding variable data areas. The appearance information is applied to variable data areas by print servers or digital front ends running RIP software from HP or a third party – for example the Harlequin software provided by Global Graphics or similar software from HP, Creo, or Esko installed on HP's print servers or digital front end computers. *See, e.g.*, Ref. [10] at 7; Ref. [13] at 2. VDP files provide appearance information to correspond with the variable data areas. For example, PDF and PDF/VT files include resource objects, XObjects, and ExtGState objects that define the graphics state and text state for variable data areas. Ref. [16] at §§ 4.3, 5.2. The graphics state includes, for example, a current transformation matrix that defines rotation and skew associated with a variable data area, color information, text characteristics including font, font size, and line characteristics. Ref. [16] at §§ 4.3, 5.2. In another example, in PPML files, the MARK element and the elements it encloses collectively

‘233 Patent, Claim 14	
	define the appearance of the object to be marked. Appearance information includes format, dimensions and clipping box (optional). The format attribute indicates the format of the data (e.g., PostScript, PDF, TIFF, etc.). The dimension attribute includes the dimensions of a rectangle that encloses the content data contained in the Source element. The clipping box attribute supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data. The PPML specification explains as follows: “The MARK element specifies the actual placement of marks on a page. It is used either for the placement of Objects (section 5.7) or for placing an Occurrence of a Reusable Object (section 5.12). The Consumer places MARKs on a page in the order in which they are listed in the PAGE element. MARKs later in a PAGE element are placed on top of the earlier ones.” Ref. [11] at 22; Ref. [12] at 34. “The VIEW element combines a TRANSFORM with a CLIP_RECT to form a description of how a particular set of content data is to be rendered…VIEW can occur in MARK, OBJECT, REUSABLE_OBJECT and OCCURRENCE.” Ref. [11] at 24; Ref. [12] at 36. “The TRANSFORM element represents a two-dimensional homogeneous transformation matrix…TRANSFORM can occur in VIEW.” Ref. [11] at 25; Ref. [12] at 37. “The OBJECT element associates a VIEW with a SOURCE to specify the clip, scale and orientation of an item of appearance data within a MARK or a REUSABLE_OBJECT.” Ref. [11] at 27; Ref. [12] at 39. “The SOURCE element defines a set of one or more content elements (EXTERNAL_DATA, INTERNAL_DATA), of a single format, to be collected into a single sequence of appearance data. The content data from all enclosed elements are concatenated in the order the elements appear, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the Consumer as a single object.” Ref. [11] at 28; Ref. [12] at 40.

<table>
<tr><th>'233 Patent, Claim 14</th><th></th></tr>
<tr><td></td><td>

<table>
<tr><th>Attribute</th><th>Required /Optional</th><th>Type</th><th>Description</th></tr>
<tr><td>Format</td><td>Required</td><td>Keyword</td><td>Indicates format of the data (e.g., PostScript, PDF, TIFF, etc.). Value: any format name registered with the Internet Assigned Numbers Authority (IANA).[4]</td></tr>
<tr><td>Dimensions</td><td>Required</td><td>Number ×2</td><td>The width <i>w</i> and height <i>h</i> of a rectangle that encloses the content data contained in this element. See 5.8.5, "Dimensions and ClippingBox" below.</td></tr>
<tr><td>ClippingBox</td><td>Optional</td><td>Number ×4</td><td>Supplies the coordinates of the lower left and upper right corners of the rectangle containing the desired area of the content data, in PPML default coordinates.</td></tr>
</table>

Ref. [11] at 28; Ref. [12] at 40.

In another example, PPMLT files provide a variety of appearance information such as spacing, size, location, font, word spacing, letter spacing, justification, and color for variable data. The appearance information appears within XSLT scripts embedded in the PPMLT file, e.g., <svg:text x="82.5pt" y="10pt" font-family="Helvetica" fontsize="10pt" word-spacing="1.294pt" letter-spacing=".129pt" text-anchor="middle" fill="rgb(255,255,255)">. Ref. [10] at 46.

In yet another example, JLYT files provide a variety of appearance information. JLYT format is the HP press's proprietary format, and allows for the full use of HP Indigo Press features and optimization. Ref. [14] at 17. JLYT files include "channels", which define the position, scaling, and rotation of separately defined "content packages." Ref. [15] at 4. JLYT files also incorporate image rules that can alter appearance information such as font, color, size, or content of fixed text or variable text fields. See Ref. [14] at 16.

HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
</td></tr>
<tr><td>(f) merging the first variable data bitmap with a copy of the static bitmap to produce a first output bitmap;</td><td>
HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to merge the variable data bit map with the template bit map. <i>See</i> Ref. [13] at 2. VDP files such as PDF, PDF/VT, PPML, PPMLT, and JLYT files provide information about how to combine the variable bitmap and the template bitmap.

For example, PDF and PDF/VT allow the RIP software to merge re-used graphical elements with
</td></tr>
</table>

'233 Patent, Claim 14	
	the variable elements of the page to create final printed images that are unique for each recipient. Ref. [13] at 4-5. In another example, "PPML constructs a page image by placing a series of Marks on the page. Marks can consist of graphics, text and/or images defined in some external content data format. A Mark can reference either non-reusable or reusable content data. Reusable content data are data which may have multiple occurrences in a PPML page, document, job, dataset or environment. The PPML code defines the data as reusable, which permits the PPML consumer to cache these items in some format which may permit highly efficient reproduction." Ref. [11] at 21; Ref. [12] at 33. PPMLT files use the same tags as PPML files, and any data referenced through XSL scripting is merged via the same techniques as applied to PPML files. Ref. [10] at 9-10. In another example, JLYT files define "channels" that identify the location and orientation of content for a given printed page. Ref. [15] at 4-5. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
(g) generating a next variable data bitmap of a next one of the variable data items utilizing a graphics state associated with the variable data area;	HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused for each instance of the document, according to the contentions with respect to element (e). Appearance information is reused for each instance of the document. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. HP, Fort Dearborn, and HP's other customers run software on dedicated print servers or digital front ends, as described above, to apply the appearance information contained in the VDP file to the variable data for each instance of the document. The print servers or digital front ends create multiple variable data bitmaps, but the appearance information and the template bitmap is reused

'233 Patent, Claim 14	
	for each instance of the document. The print servers, digital front ends, or the press applies the appearance information contained in the VDP file to the variable data for each instance of the document. Multiple variable data bitmaps are created in this manner. The appearance information and the template bitmap is reused for each instance of the document. As described above, the static data bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. To render each additional variable data record, the print server or digital front end applies the appearance information to each variable data area defined in the VDP file. PDF and PDF/VT include separate objects to define each variable data area within the document. Documents include pages for each recipient, with one or more variable data areas related to each recipient. "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to refer back to previously generated template bitmaps for those objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects, making it possible to generate multiple variable data bit maps without the need to re-interpret the file. *E.g.*, Ref. [13] at 5. In addition, PDF/VT files include DPart objects and document part metadata that provide information to the RIP software so that the RIP software does not need to re-interpret the graphics state and template information on each additional page. PPML, as another example, uses a separate DOCUMENT tag to represent each instance of the document. The document instances each contain tags as described above that identify one or more variable data records. Each of these must go through the steps of reserving, retrieving, associated, and applying before they are able to be merged with the static bitmap. Ref. [11] at 15. PPMLT is structured similarly to PPML except the DOCUMENT data is dynamically created through an XSLT script embedded in the PPMLT file. For each variable data area present in a PPMLT file, an embedded XSLT "for-each" command provides the additional variable data. Ref. [10] at 45 and 54. In yet another example, JLYT files refer to external variable data that is loaded separately to the print server or digital front end. On information and belief, processing the external variable data

'233 Patent, Claim 14	
	causes the print server or digital front end to repeat the above mentioned steps for each piece of variable data in order to be merged with the static bitmap. Ref. [15] at 4. HP, Fort Dearborn, and HP's other customers may use other VDP file types with infringing characteristics, features, and functions similar to those described above in these exemplary file types.
and (h) merging the next variable data bitmap with a copy of the static bitmap to produce a next output bitmap;	The print server or by a digital front end merges the variable data bitmaps with the template bitmap according to the contentions with respect to element (f). The appearance information and the template bitmap are reused for each instance of the document. The template bitmap is only rendered once, while the variable data bitmaps must be generated for each variable data area in the subsequent documents. The template bitmap is saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. As described above, the static bitmap is saved for reuse in subsequent Pages, Documents, Jobs, and Datasets. By identifying reusable elements, the VDP file makes it possible for the RIP software to store the template bitmap. [13] at 3, 5. "Typically, this improves efficiency by avoiding two redundant burdens on the system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. PDF and PDF/VT include "Do" statements refer back to XObjects that define objects that are used repeatedly, allowing the RIP software to store the rendered objects. Alternatively, the RIP software identifies patterns of repeating objects in the PDF file and stores a template bitmap associated with the repeating objects. *E.g.*, Ref. [13] at 5. For example, the PPML specification explains that "An important resource in PPML is the Reusable Object. . . . [A] reusable piece of page content is expressed as an OCCURRENCE of a REUSABLE_OBJECT element and is accessed using OCCURRENCE_REF. This construct is central to PPML's productivity improvement." Ref. [11] at 11; Ref. [12] at 13. "The reusability feature (enabled by elements such as REUSABLE_OBJECT and SOURCE) allows the data for a picture (or any other page content) to be sent once to the Consumer, where it can be RIPped (prepared for imaging on pages) and saved (cached) for reuse in subsequent Pages, Documents, Jobs, and Datasets. Typically, this improves efficiency by avoiding two redundant burdens on the

'233 Patent, Claim 14	
	system: redundant downloading and redundant computation of the content's appearance." Ref. [11] at 11; Ref. [12] at 13. In a further example, with respect to PPMLT documents, "PPML Templating involves downloading as much as possible of a personalized print project before the production run begins. PPML itself offers significant efficiencies in file size, and templating carries it even further: it takes advantage of the fact that for many print projects, much of the print stream is repetitive and can be stored in the digital printing press (the PPML Consumer)." Ref. [10] at 7. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2. IPT believes that JLYT files similarly cache a bitmap representation of the static data area, based on the inherent efficiency of this approach, and in light of the fact that each of the objects – both static and variable – are converted into a bitmap format prior to being assembled at the print server or digital front end. *See* Ref. [15] at 5. The static bitmap and the variable data bitmap are stitched together to generate a merged document bitmap. *See* Ref. [13] at 2.
and (i) repeating steps (g) (h) for remaining variable data items in the plurality of variable data items.	The activities performed for steps (g) and (h) are repeated for each remaining variable data item in the plurality of data items.

Document Designated Attorneys' Eyes Only: Redacted Pursuant to Protective Order

Appendix Pages A0948-1180

Exhibit 18
to Maloney Declaration

From: Maness, Audrey <audrey.maness@weil.com>
Sent: Thursday, September 08, 2016 6:50 PM
To: IPT Team
Cc: HP IPT WGM Service; Brett Johnson (johnson@fr.com); Michael Rueckheim (Rueckheim@fr.com); ben-ezra@fr.com
Subject: In re IPT: verifications
Attachments: O'Neil verification.pdf; FDC Verification.pdf; IPT FDC Supplemental Interrogatory Response MDL 2.pdf

Counsel,

Attached is the verification for O'Neil and the verification for FDC, as well as a supplemental response for FDC.

Best,

Audrey

Weil

Audrey Maness

Weil, Gotshal & Manges LLP
700 Louisiana, Suite 1700
Houston, TX 77002-2755
audrey.maness@weil.com
+1 713 546 5317 Direct
+1 713 224 9511 Fax

The information contained in this email message is intended only for use of the individual or entity named above. If the reader of this message is not the intended recipient, or the employee or agent responsible to deliver it to the intended recipient, you are hereby notified that any dissemination, distribution or copying of this communication is strictly prohibited. If you have received this communication in error, please immediately notify us by email, postmaster@weil.com, and destroy the original message. Thank you.

Fort Dearborn Company's Responses and Supplements to Industrial Print Technologies LLC's Interrogatories were prepared by counsel for Fort Dearborn Company. The Answers are correct to the best of my knowledge, information, and belief.

DATED: September 8, 2016

By: /s/ T Nicholson

Tim Nicholson

Document Designated Attorneys' Eyes Only: Redacted Pursuant to Protective Order

Appendix Pages A1184-1197

Exhibit 20
to Maloney Declaration

From: Maness, Audrey <audrey.maness@weil.com>
Sent: Friday, September 23, 2016 5:29 PM
To: Alison A. Richards; Dalmau-Jones, Alexa; IPT Team; tom@nelbum.com
Cc: HP IPT WGM Service
Subject: RE: verified interrogatory responses from Cenveo
Attachments: Scan1362.pdf

Alison,

Cenveo's verification is attached.

Best,

Audrey

From: Alison A. Richards [mailto:ARichards@fitcheven.com]
Sent: Thursday, September 22, 2016 1:48 PM
To: Maness, Audrey; Dalmau-Jones, Alexa; IPT Team; tom@nelbum.com
Cc: HP IPT WGM Service
Subject: RE: verified interrogatory responses from Cenveo

Counsel,

Given the minimal amount of infringement of the VD patents by Ft. Dearborn, Cenveo, and O'Neil, as represented in their discovery responses provided after the Court's Orders on the motions to compel, we are inclined to now formally dismiss the VDP claims against them. However, Cenveo has not yet verified its interrogatory responses and, given the lack of documentary evidence produced, we are not comfortable dismissing our VDP claims against Cenveo until we receive that verification.

Can you please send that over shortly?

Thank you,
Alison

The information contained in this email message is intended only for use of the individual or entity named above. If the reader of this message is not the intended recipient, or the employee or agent responsible to deliver it to the intended recipient, you are hereby notified that any dissemination, distribution or copying of this communication is strictly prohibited. If you have received this communication in error, please immediately notify us by email, postmaster@weil.com, and destroy the original message. Thank you.

Cenveo's Responses and Supplements to Industrial Print Technologies LLC's Interrogatories were prepared by counsel for Cenveo, Inc. The Answers are correct to the best of my knowledge, information, and belief.

DATED: September 22, 2016 By: /s/ ______________________________

Document Designated Attorneys' Eyes Only: Redacted Pursuant to Protective Order

Appendix Pages A1201-1317

Exhibit 27
to Maloney Declaration

From: Maness, Audrey <audrey.maness@weil.com>
Sent: Thursday, September 08, 2016 6:50 PM
To: IPT Team
Cc: HP IPT WGM Service; Brett Johnson (johnson@fr.com); Michael Rueckheim (Rueckheim@fr.com); ben-ezra@fr.com
Subject: In re IPT: verifications
Attachments: O'Neil verification.pdf; FDC Verification.pdf; IPT FDC Supplemental Interrogatory Response MDL 2.pdf

Counsel,

Attached is the verification for O’Neil and the verification for FDC, as well as a supplemental response for FDC.

Best,

Audrey

Weil

Audrey Maness

Weil, Gotshal & Manges LLP
700 Louisiana, Suite 1700
Houston, TX 77002-2755
audrey.maness@weil.com
+1 713 546 5317 Direct
+1 713 224 9511 Fax

The information contained in this email message is intended only for use of the individual or entity named above. If the reader of this message is not the intended recipient, or the employee or agent responsible to deliver it to the intended recipient, you are hereby notified that any dissemination, distribution or copying of this communication is strictly prohibited. If you have received this communication in error, please immediately notify us by email, postmaster@weil.com, and destroy the original message. Thank you.

O'Neil Data Systems' Responses and Supplements to Industrial Print Technologies LLC's Interrogatories were prepared by counsel for O'Neil. The Answers are correct to the best of my knowledge, information, and belief.

DATED: September 2, 2016 By: /s/

Additional Appendix Materials

Document Designated Attorneys' Eyes Only: Redacted Pursuant to Protective Order

Appendix Pages A1322-1488

**IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION**

INDUSTRIAL PRINT TECHNOLOGIES, LLC, *Plaintiff,* v. O'NEIL, INC. AND HEWLETT PACKARD COMPANY	**THIS DOCUMENT RELATES TO ALL CASES** Case No. 3:15-cv-00165-M
O'NEIL DATA SYSTEMS, INC. AND HEWLETT PACKARD COMPANY	Case No. 3:15-cv-01100-M
O'NEIL DATA SYSTEMS, INC. AND HEWLETT PACKARD COMPANY	Case No. 3:15-cv-01101-M
QUAD/GRAPHICS, INC. AND HEWLETT PACKARD COMPANY	Case No. 3:15-cv-01103-M
O'NEIL DATA SYSTEMS, INC. AND HEWLETT PACKARD COMPANY	Case No. 3:15-cv-01104-M
VISTAPRINT U.S.A., INC. AND HEWLETT PACKARD COMPANY	Case No. 3:15-cv-01106-M
FORT DEARBORN COMPANY AND HEWLETT PACKARD COMPANY, *Defendants.*	Case No. 3:15-cv-01195-M

DEFENDANTS' NOTICE OF DEPOSITION TO INDUSTRIAL PRINT TECHNOLOGIES, LLC <u>UNDER FED. R. CIV. P. 30(B)(6) REGARDING FINANCIAL MATTERS</u>

PLEASE TAKE NOTICE that pursuant to Rule 30(b)(6) of the Federal Rules of Civil Procedure, Defendants will take the deposition of Plaintiff Industrial Print Technologies, LLC ("IPT"), concerning the topics set forth in Exhibit A. The deposition will begin at 9:00 am on June 24, 2016 at Weil, Gotshal & Manges LLP, 201 Redwood Shores Parkway, Redwood Shores CA, 94065 or at such other date, time and place as may be agrees upon by counsel for the parties.

The deposition will continue from day to day, excluding weekends and holidays, until completed. The deposition will be taken before a Notary Public, shall be taken by stenographic means, and may be videotaped.

Pursuant to Federal Rules of Civil Procedure 30(b)(2) and 34, Plaintiff is requested to produce all documents and things relevant to the subject matter listed in Exhibit A at or before the deposition. "Documents and things" as used herein have the same meaning as those terms do in Rule 34.

Dated: June 16, 2016

WEIL, GOTSHAL & MANGES LLP

*/s/ Edward R. Reines*________________
Edward R. Reines
edward.reines@weil.com
Amanda Branch
amanda.branch@weil.com
WEIL, GOTSHAL & MANGES, LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

Audrey L. Maness, Bar No. 24060219
audrey.maness@weil.com
WEIL, GOTSHAL & MANGES LLP
700 Louisiana, Suite 1700
Houston, TX 77002
Telephone: (713) 546-5000
Facsimile: (713) 224-9511

Attorney for Defendant-Counterclaimant
Hewlett-Packard Company

EXHIBIT A TO
DEFENDANTS' NOTICE OF DEPOSITION
OF CHARLES RAASCH

DEFINITIONS

1. "IPT," "Plaintiff," "You," and "Your(s)" mean Industrial Print Technologies LLC and all related entities, predecessors, successors, subsidiaries, divisions, and/or affiliates thereof (specifically including, but not limited to, Forrest P. Gauthier (and any entities at any time owned by or affiliated with Forrest P. Gauthier, including but not limited to Image Sciences, Anser Technology, and NBS Southern, Inc.), Varis Corporation, Tesseron Ltd., Cincinnati Print, Inc., and Acacia Research Corporation), and any and all past and present officers, directors, agents, employees, consultants, accountants, attorneys, representatives, and other persons or entities acting or purporting to act on behalf of any of the foregoing.

2. "Defendants" shall mean the defendants in the MDL actions, case number 3:15-MD-02614-M, including Hewlett-Packard Company, O'Neil Data Systems, Inc., Cenveo, Inc., Quad/Graphics, Inc., Vistaprint U.S.A., Inc., Fort Dearborn Company, and all related entities, predecessors, successors, subsidiaries, divisions and/or affiliates thereof, past or present, and all past or present officers, directors, agents, employees, consultants, accountants, attorneys, representatives, and any other person or entity acting on behalf of any of the foregoing.

3. "Lawsuit" means the MDL actions, case number 3:15-MD-02614-M.

4. "Acacia Entities" means Acacia Patent Acquisition LLC, Acacia Research Corporation, Acacia Research Group, LLC, and/or Acacia Global Acquisition LLC, and all related entities, predecessors, successors, subsidiaries, divisions and/or affiliates thereof, past or present, and all past or present officers, directors, agents, employees, consultants, accountants, attorneys, representatives, and any other person or entity acting on or purporting to act on behalf

of any of the foregoing.

5. The term “third party” means and includes any person, persons, entity or entities other than Defendants and IPT.

6. “Patents-in-Suit” or “Asserted Patents” shall mean the patents asserted in the MDL actions, case number 3:15-MD-02614-M, including United States Patent No. 5,729,665 (the “665 patent”), U.S. Patent No. 5,937,153 (the “153 patent”), U.S. Patent No. 7,274,479 (the “479 patent”), U.S. Patent No. 7,333,233 (the “233 patent), U.S. Patent No. 6,145,946 (the “946 patent”), U.S. Patent No. 6,493,106 (the “106 patent”), U.S. Patent No. 6,381,028 (the “028 patent”), including all associated applications filed in the patent office.

7. “Related Patents” means any patents and patent applications related to the Asserted Patents, including without limitation those patents and patent applications that claim priority to any one or more of the Asserted Patents.

8. The term “Accused Products” means any product made, used, sold, or offered for sale by any of Defendants that IPT contends infringes any claim of the Patents-in-Suit.

9. “Person” shall mean any natural person, organization, firm, corporation, partnership, sole proprietorship, or other legal entity, and the acts “of a Person” include the acts of owners, directors, officers, members, employees, agents, attorneys, representatives, and any other Persons acting on a Person’s behalf.

10. “Communication” shall mean any form of oral or written interchange, whether in person, by telephone, by facsimile, by telex, by mail, by electronic mail, or by any other medium.

11. “Document” shall have the meaning set forth in Federal Rule of Civil Procedure 34, and shall include without limitation information stored in electronic, magnetic, or optical

media, drafts, all translations of documents, and all materials relating to communications. A draft or non-identical copy is a separate Document within the meaning of the term.

12. "Thing" shall mean, consistent with the comprehensive meaning in Federal Rule of Civil Procedure 34, any physical specimen or tangible item, other than a Document.

13. "Information" means information in any form, including but not limited to documentary, electronic, graphical, or tabular form, and communicated by any means, including but not limited to orally, in writing, or electronically.

14. The terms "refer," "relate," or "relating to," mean pertaining to, referring to, and/or relating to the matter specified.

INSTRUCTIONS

Each of these instructions is incorporated into each topic to which it pertains:

Each topic shall operate and be responded to independently, and unless otherwise indicated, no request limits the scope of any other topic.

If you find the meaning of any term in these topics to be unclear, then you are to assume a reasonable meaning, state what the assumed meaning is, and respond to the topic on the basis of that assumed meaning.

Defendants seek discovery of your knowledge of facts concerning the topics listed below. Your knowledge of facts relating to a third party or of acts taken by or on behalf of a third party are discoverable and within the scope of permissible discovery.

Defendants do not seek discovery of information protected by the attorney-client privilege or work product immunity. The parties, however, may disagree as to what information is protected by either the attorney-client privilege or the work product immunity. Therefore, any objections to the topics listed below based on attorney-client privilege or work product immunity should be reserved and addressed on a question-by-question basis during the deposition.

TOPICS FOR DEPOSITION

TOPIC NO. 1: Your and/or the Acacia Entities' policies for licensing patents, including without limitation, the circumstances under which you will grant a license, the factors considered when deciding what terms to incorporate in a license, the structure(s) that will be considered for a license, and the amounts and royalty rates that you have offered and/or accepted for a license.

TOPIC NO. 2: The content, scope, and meaning of agreements concerning rights in or licenses to the Patents-in-Suit.

TOPIC NO. 3: The facts and circumstances surrounding all patent licenses related to any of the Patents-in-Suit, including without limitation, the persons involved in the negotiation of the license; the persons involved in the ultimate decision to execute the license; the persons involved in the execution of the license; the terms, nature, and scope of the licenses; whether the licenses contemplate lump sum payments or running royalties; the stated or implied royalty rate; the duration of the licenses; whether the licenses are restricted in any way; and the identity and location of documents sufficient to show the foregoing.

TOPIC NO. 4: Each product made, offered for sale, or sold under license from you that is covered by an asserted claim of the Patents-in-Suit, together with an identification of the claim(s) that cover the product; the period in which the product was under development, manufactured, offered for sale, and sold; and the quantities in which each product was made and sold by or for your licensee.

TOPIC NO. 5: The indexing, tracking, and monitoring by you of any products allegedly embodying the invention(s) of the Patents-in-Suit.

TOPIC NO. 6: All facts and circumstances relating to your efforts to satisfy the marking and notice requirements of 35 U.S.C. § 287 with respect to the Patents-in-Suit.

TOPIC NO. 7: Every instance where you asserted or threatened to assert any of the Patents-in-Suit or any other Related Patents against any entity other than the named Defendants in this Lawsuit, including all facts and circumstances leading up to the assertion or threatened assertion and all facts and circumstances resulting from the assertion or threatened assertion.

TOPIC NO. 8: Your first knowledge or awareness of Defendants' alleged infringing activity, including without limitation, your first knowledge or awareness of any of the Accused Products.

TOPIC NO. 9: The facts and circumstances concerning any efforts by you to provide notice to or otherwise communicate with Defendants concerning the Patents-in-Suit or Related Patents before filing this Litigation.

TOPIC NO. 10: The reasons why you and your predecessor(s)-in-interest did not file an action for infringement of the Patents-in-Suit against Defendants before September 12, 2014.

TOPIC NO. 11: Any economic analysis or valuation of you or the Patents-in-Suit performed by or on behalf of you, any predecessor company, or any third party.

TOPIC NO. 12: Any analysis, communications, or documents you contend may be related to determining a reasonable royalty for the Patents-in-Suit.

TOPIC NO. 13: All actual damages that you claim to have sustained as a result of Defendants' alleged infringement.

TOPIC NO. 14: Any value assigned to any of the Asserted Patents, Related Patents, or related applications (either alone, or together with other patents or consideration, including as part of an intellectual property portfolio) by you, any predecessor company, or any third party for purposes of any internal or external accounting purpose, merger, acquisition, divestiture, or any other business transaction.

TOPIC NO. 15: Any valuations of technology or patents similar to, relevant to, or covering related subject matter to the Asserted Patents, the Related Patents, or related applications.

TOPIC NO. 16: All facts and circumstances relating to your contentions that Defendants have willfully infringed the Patents-in-Suit.

TOPIC NO. 17: All facts and circumstances regarding individuals, companies, and other entities that provide financing to you or that are entitled to revenue related to the Asserted Patents, Related Patents, or this Litigation

TOPIC NO. 18: All details concerning any portion, share, or interest in the Asserted Patents or Related Patents sold or purchased by you, including all such transactions between IPT and the Acacia Entities.

TOPIC NO. 19: All offer(s) of value, tangible or intangible, monetary or otherwise for any of the Patents-in-Suit that you are aware of, including without limitation, the identity of the entity making the offer(s); the persons involved in any negotiations related to the offer(s); and the nature and scope of the offer(s).

TOPIC NO. 20: Your composition, ownership, and operation, including without limitation, an identification of all debt holders, investors, shareholders, officers, directors, managing agents, parents, subsidiaries and affiliated companies.

TOPIC NO. 21: All facts and circumstances concerning the creation, formation, growth, decline, revenues, costs, profits, debts, audits, and governance of IPT.

TOPIC NO. 22: Your assets and liabilities.

TOPIC NO. 23: Your business plans, forecasts, and/or projections of sales, profits, or licensing or royalty revenues related to the subject matter of the asserted claims of the Patents-in-Suit.

TOPIC NO. 24: The identity and location of all persons with knowledge of any of the foregoing topics and your efforts to locate them.

CERTIFICATE OF SERVICE

I hereby certify that on June 16, 2016 a true and correct copy of the foregoing document was served by email on the recipients below:

FITCH, EVEN, TABIN & FLANNERY LLP
Timothy P. Maloney
tpmalo@fitcheven.com
Alison Aubry Richards
ARichards@fitcheven.com
Nicole L. Little
nlittle@fitcheven.com
David A. Gosse
dgosse@fitcheven.com
Steven C. Schroer
scschr@fitcheven.com

NELSON BUMGARDNER PC
Thomas M. Cecil
tom@nelbum.com

Attorneys for Plaintiff
Industrial Print Technologies LLC

/s/ Andrew Greenfeld
Andrew Greenfeld

Document Designated Attorneys' Eyes Only: Redacted Pursuant to Protective Order

Appendix Pages A1498-1502

GLOBAL GRAPHICS : case study

HIGH SPEED AT THE RIGHT PRIc E

INDUSTRY SECTOR

Digital Production Printing

ISSUE

Digital Front Ends (DFEs) for digital presses need to cope with rapidly increasing data processing requirements as print jobs become more complex. At the same time the need for speed is an on-going challenge for press manufacturers: the DFE must process data quickly enough to keep up with the press and to drive it at full rated speed. If you're a press manufacturer how do you minimize the costs of your DFE compared with the cost of the press?

GLOBAL GRAPHICS' PRODUCT

Harlequin RIP®: processes PostScript®, PDF and XPS natively in one RIP engine.

SOLUTION

Global Graphics has worked with HP to minimize the total costs of the DFE through optimization of the RIP software. A close, collaborative development partnership, whereby each party shares product roadmaps early, plays a crucial part in the success of projects.

APPLICATION

The Harlequin RIP is the engine that drives the HP Indigo digital press range via either the HP SmartStream Production Pro Print Server or the HP SmartStream Ultra Print Server.

The presses that can be driven using the Harlequin RIP powered DFE are:

HP Indigo 7500 Digital Press
HP Indigo 7000 Digital Press
HP Indigo W7200 Digital Press
HP Indigo press 5500
HP Indigo press 5000
HP Indigo press ws4500
HP Indigo press ws4050
HP Indigo 3550 Digital Press
HP Indigo press 3500
HP Indigo press 3050
HP Indigo press w3250
HP Indigo press w3200
HP Indigo press 3000
Indigo Ultrastream
HP Indigo WS6000p Digital Press
HP Indigo press s2000

Variations on the HP DFEs are also used to drive the inkjet web presses such as the T400 from the Inkjet High-speed Production Solutions division of HP.

GLOBAL GRAPHICS™
software

In recent years the amount of data consumed by the raster image processors (RIPs) that drive digital presses has increased significantly. Graphic design jobs are more complex because of the use of transparency, which increases the amount of processing a RIP has to do. The trend toward expanded use of personalization of direct marketing pieces, as well as transpromo jobs, has also increased the load on the RIP, as has the sharp rise in demand for image-heavy jobs such as photo books.

So the RIP, often the unsung hero of the printing process, has to be more intelligent and considerably faster than it used to be. When deployed in the RIP farms or DFEs that are driving state of the art digital presses the data rate out of the DFE has to be fast enough to drive the press at rated speed. And digital presses themselves are getting faster and faster.

Market factors

Let's look at that increase in data again. There is now more variable coverage in direct marketing pieces than there ever used to be. Simultaneously, there has been a move to add richer graphics into statements and bills. In parallel with that trend a lot of transactional work has been taken out of the data center where it used to be printed with AFP and is now printed from PDF on commercial presses.

What about photo printing? Just about the hardest common combination of things you can put into a PDF file, as far as the RIP is concerned, is a high-resolution image involving transparency. Drop shadows and soft edges are really becoming quite common in some photo book applications so there are far more instances of transparency than there used to be. Fortunately the Harlequin RIP excels at processing transparency, interpreting PDF natively since 1997 and interpreting live transparency natively since 2002, removing the need to flatten files.

Optimize for what the customer needs

Given the right DFE architecture it's always possible to achieve rendering performance adequate to drive the digital press at rated speed. The challenge isn't just to be fast enough, it's to achieve that goal without incurring an uneconomically high cost for the bill of materials to build the DFE. By making the Harlequin RIP exceptionally fast and efficient Global Graphics has allowed HP Indigo to achieve engine speed with fewer copies of the RIP, with a concomitant reduction in the costs for hardware, operating systems and other associated software. And that makes it greener too, because you need fewer computers and less power for the DFE.

So how fast can the RIP be made to go when it is already a first-rate performer? Answer: focus on the biggest challenges facing your customer. In response to the increasing number of jobs containing transparency Global Graphics has, for several years, dedicated a team of programmers to optimize every line of code in the RIP. But the latest secret weapon is adding multi-threaded compositing of transparency. c ompositing is the part of the RIP that takes each transparent object and figures out what the final color has to be when you add a transparent layer on top of a sandwich of color objects. This certainly makes a big difference when processing photo book jobs with drop shadows and soft edges.

HP Indigo W7200
Digital Press

GLOBAL GRAPHICS : case study

HP Indigo 7500
Digital Press

> “ The challenge isn't just to be fast enough, it's to achieve that goal without incurring an uneconomically high cost for the bill of materials to build the DFE. ”

In the case of handling variable data, for some years now Global Graphics has been extending the 'PDF retained raster' feature in the RIP and this, together with the post-RIP stitching facilities provided by HP Indigo, has achieved some significant performance improvements. How does PDF retained raster work? "The RIP deconstructs the pages into foregrounds and backgrounds" says Martin Bailey, c TO Global Graphics, "and intelligently detects those backgrounds that are shared between multiple pages.The shared backgrounds are rendered just once – that could be once instead of 50,000 times – and as they often contain all the complex graphics on the page, the performance increase is significant. There's no hard limit on the number of backgrounds in any one job but we usually say that if 10% of the pages in a job use the same background we'll recognize it as shared.You can change that default if you want to." The reason this RIP feature is called PDF retained raster is that the background data is "retained" and returned to the RIP process when it is required, or handed off to an OEM partner's code if they've implemented that, as HP Indigo have done.

Global Graphics has also developed solutions for other aspects of the DFE besides performance. Although Harlequin has included in-RIP color management for many years, that has needed some extensions to fully optimize for the HP Indigo's specific requirements, for example.

Sharing roadmaps

To achieve performance gains fine-tuned to a particular customer requires a very close integration of support teams. "You can't just throw a finished RIP over the wall and hope it works with your customer's hardware," says Martin Bailey, "It is critical that Global Graphics involves HP Indigo as a partner in the design and delivery process. Pre-planning can be several years in advance of a new model. We discuss quite some way ahead of time feature sets that we plan to put into our next RIP version and are willing to adjust schedules, prioritization and implementation details to meet specific needs. You can only really do this effectively if you share product road maps in both directions". It's important to continue that approach all the way through a project, says Bailey, so that as Global Graphics refines its RIP design, that level of detail is provided to the partner.

"You need to keep that cooperation going all the way through because it's easier to make changes earlier in the design process." Bailey states. "We provide early access to builds so that HP can start their development. And we provide even earlier access to high-level descriptions of the functionality and APIs.This drills down to more and more precision over time as we get toward sending over software so that HP Indigo can start designing their part of the equation even before they have something to plug it into."

June 2012

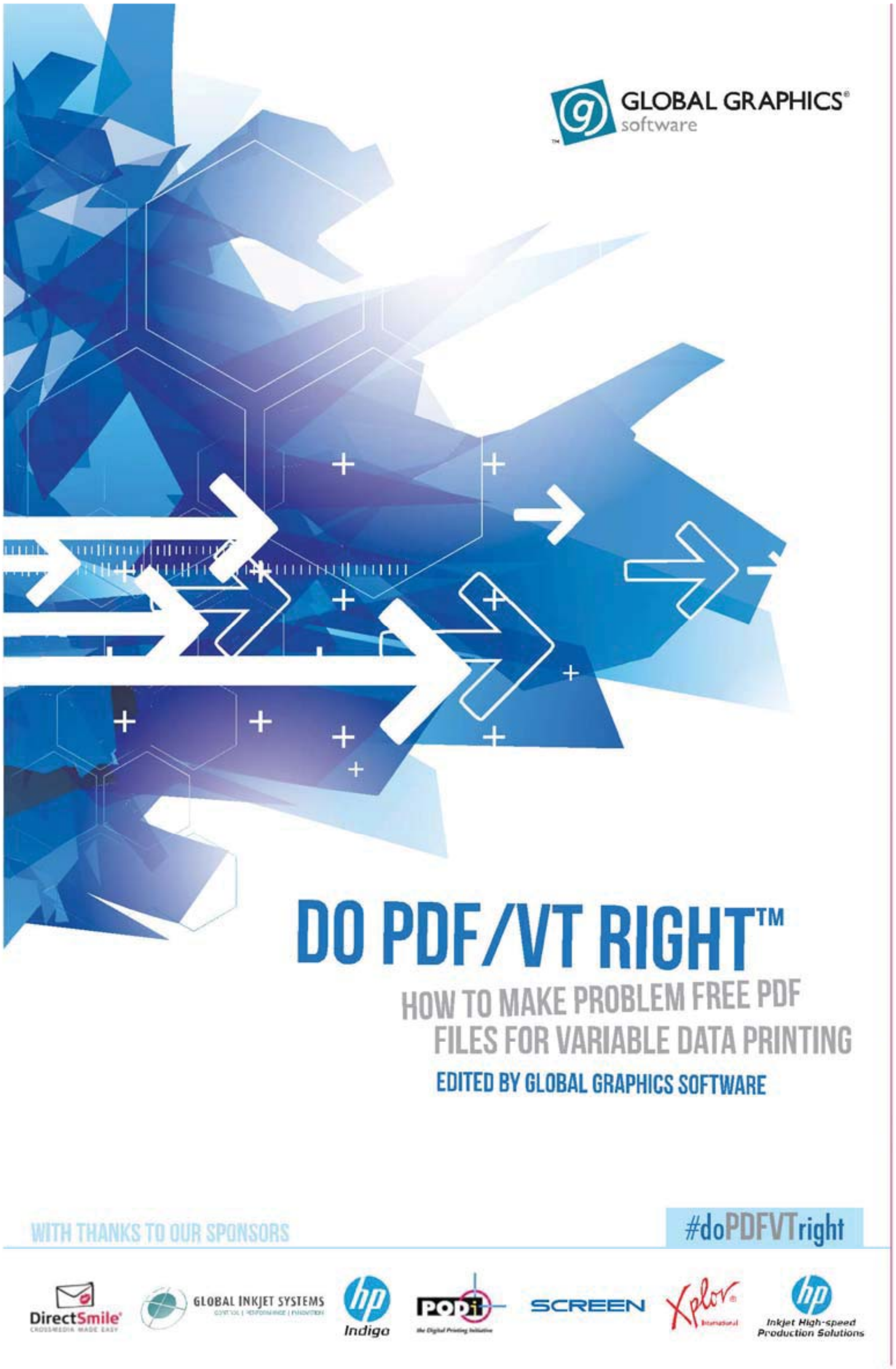
GLOBAL GRAPHICS®
software

Global Graphics Software Inc.
Somerset c ourt, Suite 320
281 Winter Street
Waltham, MA 02451, USA
Tel: +1-978-849-0011

Global Graphics Software Ltd
Building 2030
c amboume Business Park
c amboume, c ambridge
c B23 6DW UK
Tel: +44 (0)1954 283100
Fax: +44 (0)1954 283101

Global Graphics K.K.
704 AIOS Toranomon Bldg.
1-6-12 Nishi-Shimbashi,
Minato-ku, Tokyo
105-0003, Japan
Tel: +81-3-6273-3740
Fax: +81-3-6273-3741

www.globalgraphics.com



GLOBAL GRAPHICS® software

DO PDF/VT RIGHT™

HOW TO MAKE PROBLEM FREE PDF FILES FOR VARIABLE DATA PRINTING

EDITED BY GLOBAL GRAPHICS SOFTWARE

WITH THANKS TO OUR SPONSORS

#doPDFVTright

DirectSmile · GLOBAL INKJET SYSTEMS · hp Indigo · PODi · SCREEN · Xplor · hp Inkjet High-speed Production Solutions

WITH THANKS TO:

DirectSmile
CROSSMEDIA MADE EASY

GLOBAL INKJET SYSTEMS

hp
Indigo

PODi
the Digital Printing Initiative

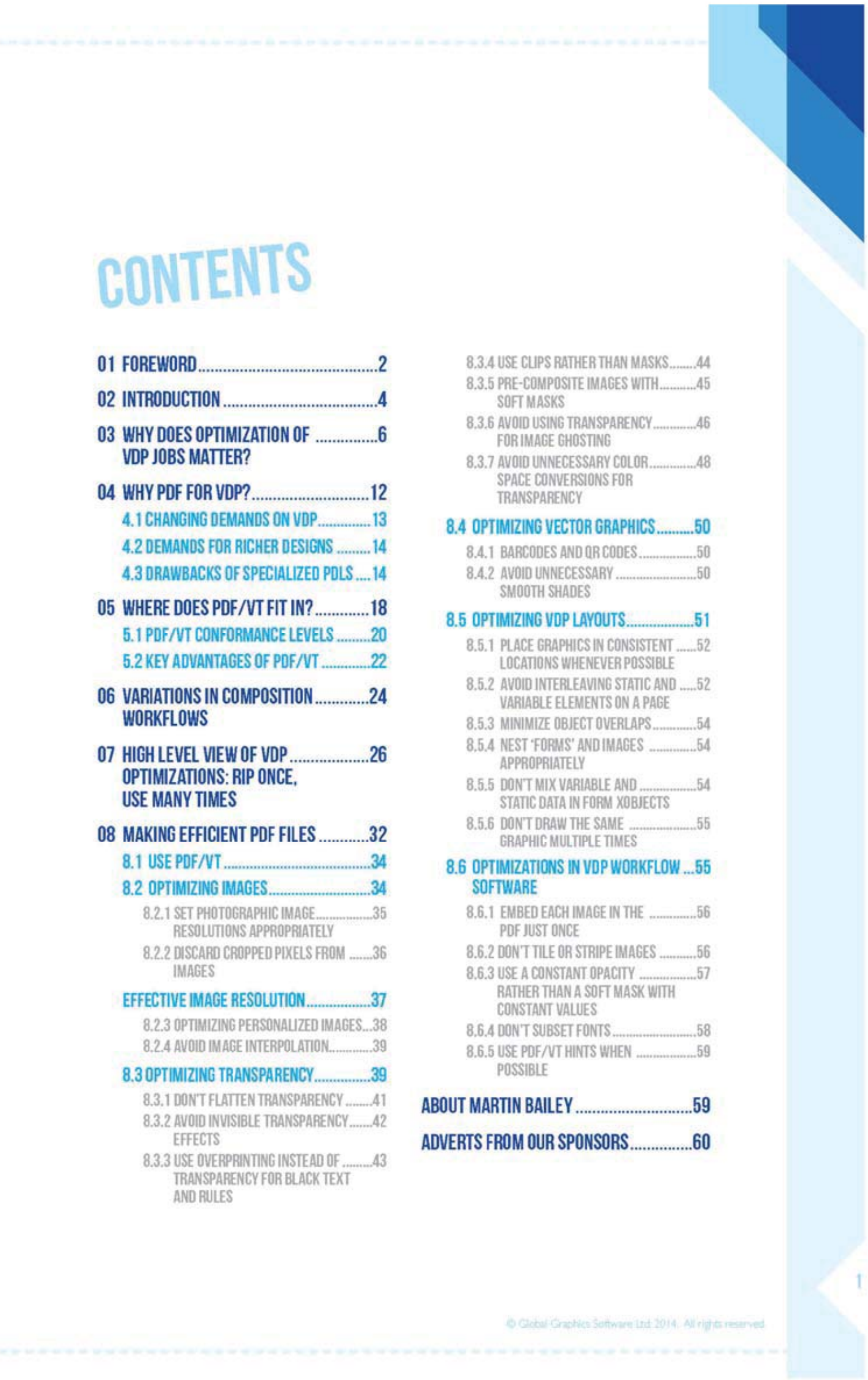
SCREEN

Xplor
International

hp
Inkjet High-speed
Production Solutions

The contents of this document are provided "AS IS" and without warranty. This document could contain technical inaccuracies, typographical errors and out-of-date information. The information contained herein may be updated or changed without notice at any time. Use of the information is therefore at your own risk. Global Graphics Software Ltd shall not be liable for technical or editorial errors or omissions contained herein.

Published by Global Graphics Software Ltd. March 2014. You may not edit or modify this document without the prior written approval of Global Graphics Software.



CONTENTS

01

FOREWORD

By almost any definition, the portable document format (PDF) is a winner: there are 585 million PDF files on the Web, there are billions and billions of PDF files in company archives, it is by far the most popular file format for static and variable print production, and it has a name recognition that many brand owners could only dream of for their products.

In the early days of PDF it was not evident that it would ever become so popular, especially in print production. Though based on PostScript®, the file format was originally developed for electronic document exchange – ironically enough to foster the paperless office. There was very little initial support for production print. For instance, the first version of PDF only had support for the RGB color space.

Over time, as adoption grew, it became clear that PDF was an excellent format for the graphic arts industry in terms of document exchange, proofing, annotation, and prepress. Some technical modifications were made to PDF to better support the professional print user, mainly around CMYK, halftones, transparency, and overprinting. Other important improvements came with the ISO PDF/X standards in the early 2000s. PDF/X restricted the flexibility of the format in return for more predictable exchanges of prepress data. Since then the use of PDF has taken off in professional print production.

Yet – despite its popularity, I wonder how many people really understand the format beyond the basic benefits. Creating well-constructed PDF files isn't a simple task and was made more difficult because accessible and applicable information on this topic has not been easy to find… until now.

Martin Bailey, the author of this guide, is one of the world's foremost PDF experts, has been integrally involved in standards efforts for file format and workflow, and is extremely knowledgeable about PDF construction, processing, and raster image processing. Besides that, I know Martin as a person who is always willing to share his 20+ years industry insights on this topic. He and Global Graphics should be applauded for their efforts to create this guide as it contains a wealth of actionable insights that will help the industry become more efficient.

I highly recommend this guide for designers, prepress experts, variable data, and other prepress software providers who work with PDF. Bigger, faster, and more productive digital print devices require amazing amounts of data. Improperly created PDFs can choke these production workflows. Therefore it is crucial for production personnel to know how they can optimize production. This guide will be a great tool for them.

Sincerely,

Kaspar Roos

Director of InfoTrends' Production Software Services
London, February 2014

INTRODUCTION

Over the last fifteen years variable data in digital printing has grown from "the next big thing" with vast, untapped potential to a commonly used process for delivering all manner of personalized information. VDP is used for everything from credit card bills and bank statements to direct mail postcards and personalized catalogues, from college enrolment packs to Christmas cards and photobooks, from labels to tickets, checks to ID cards.

This huge variety of jobs is created and managed by an equally huge variety of software, from specialist composition tools to general purpose design applications carefully configured for VDP. And they are consumed by workflows involving (or even completely within) the Digital Front End (DFE) for a digital production press, where jobs must be imposed, color managed, RIPed and processed in much the same way as a non-variable job.

And those tools are installed, adjusted and used by an equally broad variety of people. Obviously there are graphic designers, brand owners and marketing campaign managers involved, but also experts in data management and manipulation, specialists in combining that data with the graphical design, and last (but definitely not least) the operators for digital production presses and the finishing equipment that is used to cut, trim, fold and to stuff envelopes.

With that variety of use cases, creation software and print options, this means that a huge number of complex and sophisticated workflows have evolved over an amazingly short time. And most of the time they work just fine – the job is printed and delivered as designed.

Over the years a number of projects and publications have been developed to assist with designing jobs for digital print. One of the first, and probably the most influential and famous was "Design for Digital" from the Printing Industries Alliance Digital Print Council. Many colleges and universities also offer courses in how to design a job for digital print, as do some production press vendors.

The majority of those publications and courses, however, are either "how-to" guides for getting the most out of a specific piece of composition software, or are aimed at maximizing the response rate on direct mail pieces. Both of those are useful goals, but they leave unanswered the question of optimizing the internal structure of VDP jobs for efficient processing.

And that's the goal of this guide: to provide a set of actionable recommendations that help you ensure that your jobs don't slow down the print production workflow … without affecting the visual appearance that you're trying to achieve.

As a side benefit, several of the recommendations set out below will also ensure that your PDF files can be delivered more efficiently on the web and to PDF readers on mobile devices in a cross-media publishing environment.

Martin Bailey, CTO, Global Graphics Software

03

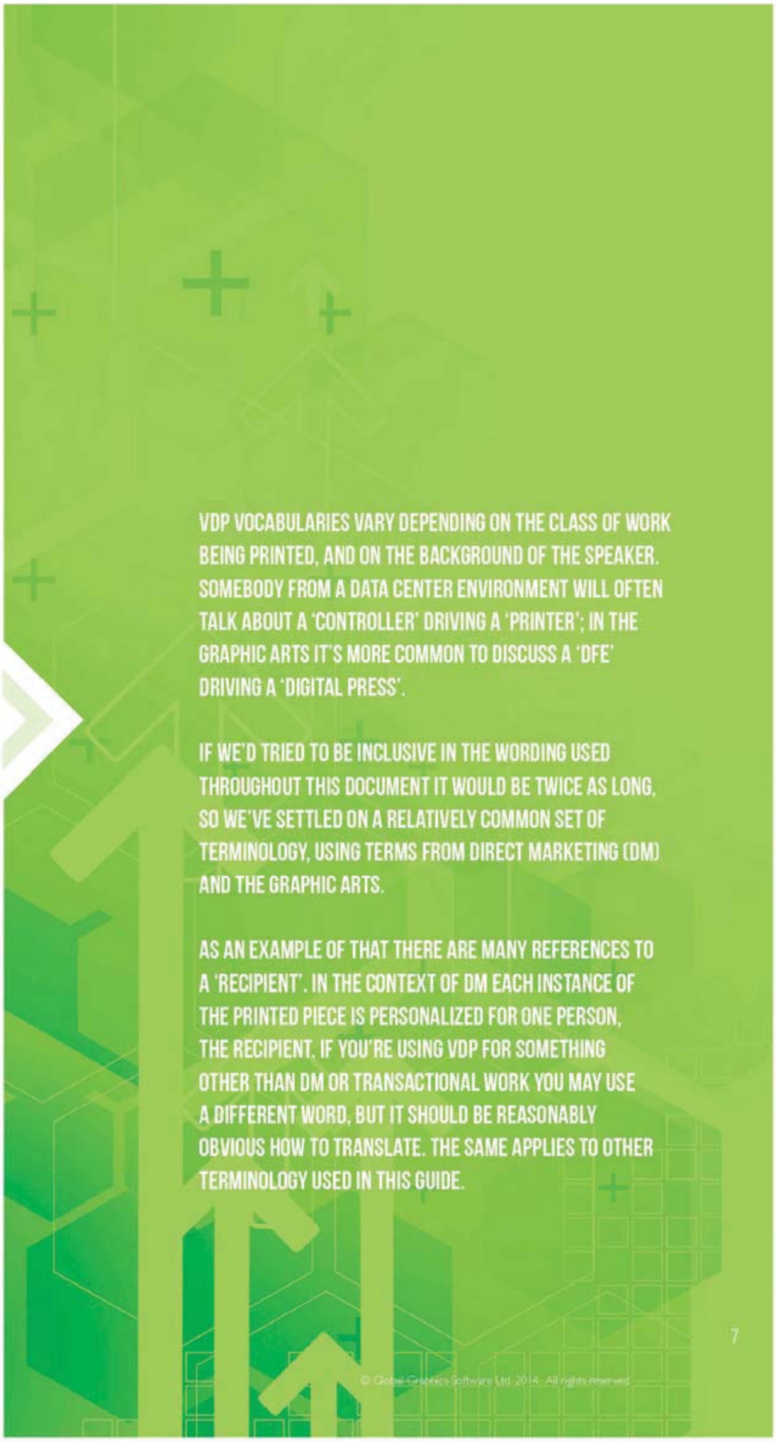
WHY DOES OPTIMIZATION OF VDP JOBS MATTER?

If you're printing commercial, publication or PoD work on a digital press you'll usually be producing short runs; if you weren't, you'd probably be using an offset or flexo press instead. But "short runs" very rarely means a single copy.

If you're printing, for example, 50 copies of a series of booklets, you only need to RIP each sheet once. To continue the example, let's assume that you're printing on a press that can produce 100 pages per minute. Assuming that all your jobs are 50 copies long, you therefore need to RIP jobs at only two pages per minute. Once a job is fully RIPed and the copies are running on press you have plenty of time to get the next job prepared before the current one clears the press.

But VDP jobs place additional demands on the processing power available in a DFE because most pages are different to every other page and must therefore each be RIPed separately. If you're printing at 100 pages per minute then you must RIP at 100 pages per minute. Because of this a variety of optimizations have been developed in DFEs that mean that parts of many pages don't need RIPing so rapidly and these are described below, but even with those optimizations a complex VDP job typically requires significantly more processing power than a 'static' job where every copy is the same.

The amount of processing required to prepare a PDF file for print in a DFE can vary hugely without affecting the visual appearance of the printed result, depending on how it is constructed. Poorly



VDP VOCABULARIES VARY DEPENDING ON THE CLASS OF WORK BEING PRINTED, AND ON THE BACKGROUND OF THE SPEAKER. SOMEBODY FROM A DATA CENTER ENVIRONMENT WILL OFTEN TALK ABOUT A 'CONTROLLER' DRIVING A 'PRINTER'; IN THE GRAPHIC ARTS IT'S MORE COMMON TO DISCUSS A 'DFE' DRIVING A 'DIGITAL PRESS'.

IF WE'D TRIED TO BE INCLUSIVE IN THE WORDING USED THROUGHOUT THIS DOCUMENT IT WOULD BE TWICE AS LONG, SO WE'VE SETTLED ON A RELATIVELY COMMON SET OF TERMINOLOGY, USING TERMS FROM DIRECT MARKETING (DM) AND THE GRAPHIC ARTS.

AS AN EXAMPLE OF THAT THERE ARE MANY REFERENCES TO A 'RECIPIENT'. IN THE CONTEXT OF DM EACH INSTANCE OF THE PRINTED PIECE IS PERSONALIZED FOR ONE PERSON, THE RECIPIENT. IF YOU'RE USING VDP FOR SOMETHING OTHER THAN DM OR TRANSACTIONAL WORK YOU MAY USE A DIFFERENT WORD, BUT IT SHOULD BE REASONABLY OBVIOUS HOW TO TRANSLATE. THE SAME APPLIES TO OTHER TERMINOLOGY USED IN THIS GUIDE.

constructed PDF files can therefore impact a print service provider in one or both of two ways:

- Output is not achieved at engine speed, reducing return on investment (ROI) because fewer jobs can be produced per shift. In extreme cases when printing on a continuous feed (web-fed) press a failure to deliver rasters for printing fast enough can also lead to media wastage and may confuse in-line or near-line finishing.
- In order to compensate for jobs that take longer to process in the DFE, press vendors often provide more hardware to expand the processing capability, increasing the bill of materials, and therefore the capital cost of the DFE.

The effect of these varies slightly with the class of digital press in use:

For a light production digital press model (with a recommended maximum monthly volume below 1M pages) you usually only get one choice of DFE. Vendors work hard to ensure that the processing power of that single model is appropriate for the majority of users, but of course this means that it's a little more expensive than required for people who only run simple jobs, and is not guaranteed to achieve engine speed for print sites handling more sophisticated jobs.

The sales team for vendors of high-volume presses (with a duty cycle over 1.5M pages per month) will often work closely with a prospective buyer to understand the mix of jobs that they want to run, the number of shifts they operate and their expectations for turn-round times on jobs. At the end of that consultation they'll recommend a particular size of DFE that's designed to ensure that the press is driven fast enough to meet the print service provider's expectations and needs without being over-specified and therefore costing more than required.

Once the press is installed and running the production manager will usually calculate and tune their understanding of how many jobs of what type can be printed in a shift. Customer services representatives work to ensure that customer expectations are set appropriately, and the company falls into a regular pattern. Most jobs are quoted on an acceptable turn-round time and delivered on schedule.

But occasionally a customer supplies a file that takes much longer than expected to process and disrupts the whole schedule. Depending on how many presses the print site has, and how they are connected to one or more DFEs this may lead to a press sitting idle, waiting for pages to print. It may also delay other jobs in the queue, or mean that they must be moved to a different press. Moving jobs at the last minute may not be easy if the presses available are not identical. Different presses may require different print streams or imposition and there may be limitations on stock availability, etc.

Many DM or transactional jobs have tight deadlines on delivery schedules; they may need to be ready for a specific time for posting, with penalties for late delivery, or the potential for reduced return for the marketing department behind a direct mail campaign.

This guide is designed to help you avoid making jobs that disrupt and delay the printing process, increasing the probability of everyone involved in delivering the printed piece hitting their deadlines reliably, and achieving their goals effectively.

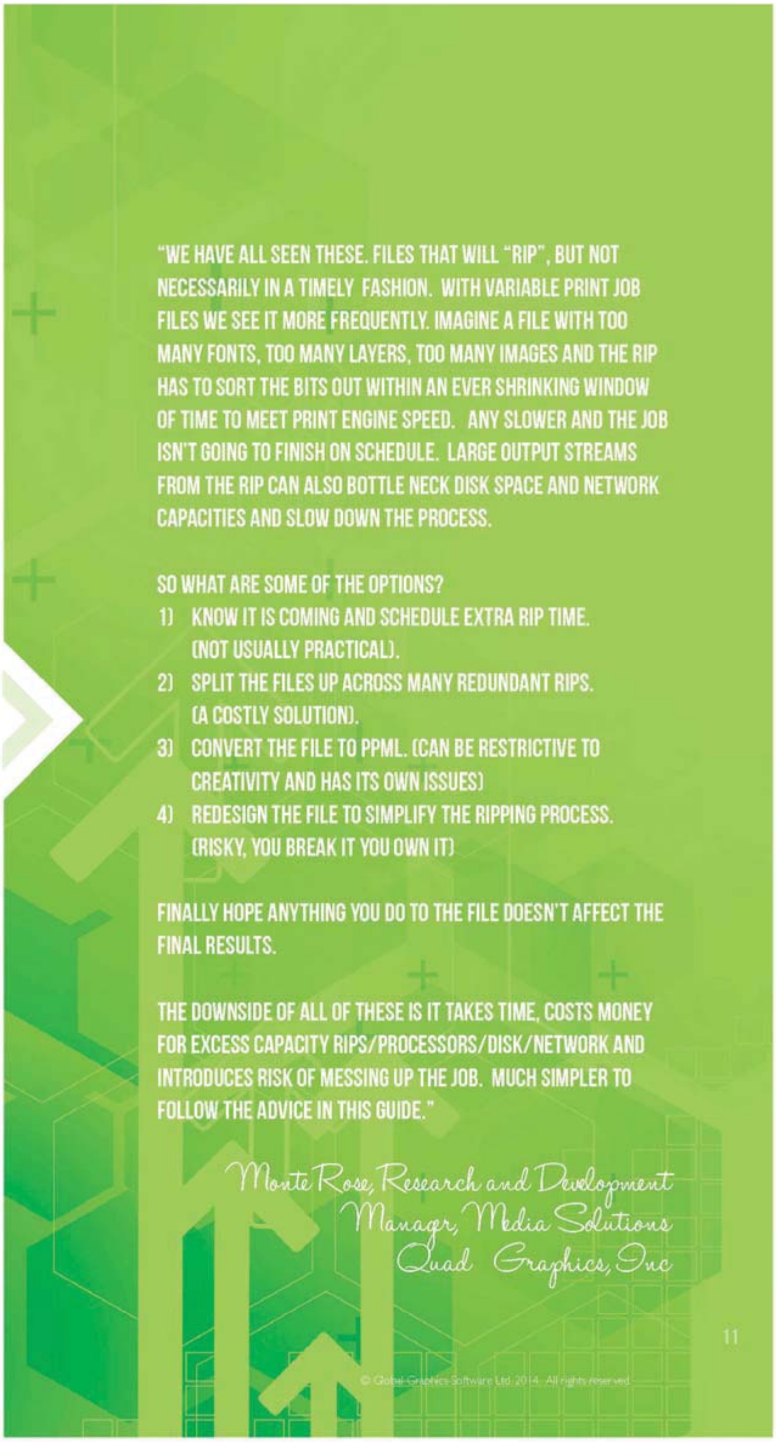
This isn't to compensate for any inadequacy of the DFEs in use with digital presses. Think of it as being similar to avoiding filling your brand new Ferrari with cheap and inferior fuel!

Each minor inefficiency in a VDP job will often only add between a few milliseconds and a second or two to the processing of each page, but those times need to be multiplied up by the number of pages in the job. An individual delay of half a second on every page of a 10000 page job adds up to over half an hour for the whole job. For a really big job of a million pages it only takes an extra tenth of a second per page to add 24 hours to the total processing time.

If you're printing at 120ppm the DFE must process each page in an average of half a second or less to keep up with the press. On the fastest continuous feed inkjet presses at 5200ppm one page must be processed every 11.5ms. It doesn't take much of a slow-down to start impacting throughput.

> "NOW THE INDUSTRY HAS CLEARLY EMBRACED THIS NEW STANDARD, PDF/VT ENABLED WORKFLOWS WILL OPTIMIZE AND EFFICIENTLY MANAGE A WIDE VARIETY OF VARIABLE DATA PRINTING JOBS"

Harry Raaphorst,
Managing Director, Direct Smile



"WE HAVE ALL SEEN THESE. FILES THAT WILL "RIP", BUT NOT NECESSARILY IN A TIMELY FASHION. WITH VARIABLE PRINT JOB FILES WE SEE IT MORE FREQUENTLY. IMAGINE A FILE WITH TOO MANY FONTS, TOO MANY LAYERS, TOO MANY IMAGES AND THE RIP HAS TO SORT THE BITS OUT WITHIN AN EVER SHRINKING WINDOW OF TIME TO MEET PRINT ENGINE SPEED. ANY SLOWER AND THE JOB ISN'T GOING TO FINISH ON SCHEDULE. LARGE OUTPUT STREAMS FROM THE RIP CAN ALSO BOTTLE NECK DISK SPACE AND NETWORK CAPACITIES AND SLOW DOWN THE PROCESS.

SO WHAT ARE SOME OF THE OPTIONS?

1) KNOW IT IS COMING AND SCHEDULE EXTRA RIP TIME. (NOT USUALLY PRACTICAL).
2) SPLIT THE FILES UP ACROSS MANY REDUNDANT RIPS. (A COSTLY SOLUTION).
3) CONVERT THE FILE TO PPML. (CAN BE RESTRICTIVE TO CREATIVITY AND HAS ITS OWN ISSUES)
4) REDESIGN THE FILE TO SIMPLIFY THE RIPPING PROCESS. (RISKY, YOU BREAK IT YOU OWN IT)

FINALLY HOPE ANYTHING YOU DO TO THE FILE DOESN'T AFFECT THE FINAL RESULTS.

THE DOWNSIDE OF ALL OF THESE IS IT TAKES TIME, COSTS MONEY FOR EXCESS CAPACITY RIPS/PROCESSORS/DISK/NETWORK AND INTRODUCES RISK OF MESSING UP THE JOB. MUCH SIMPLER TO FOLLOW THE ADVICE IN THIS GUIDE."

Monte Rose, Research and Development Manager, Media Solutions
Quad Graphics, Inc

11

© Global Graphics Software Ltd 2014. All rights reserved

04

WHY PDF FOR VDP?

This guide is specifically about optimizing PDF files for variable data print. PDF and PDF/X (a standardized subset of PDF) have become the dominant delivery formats for conventional print and print on demand (PoD) over the last decade.

In 2010 InfoTrends' End-User Workflow Survey asked the question "Please select the top two optimized print output formats used for variable data job production". The data that they collated clearly shows that the run-away winner at the top of the list was "Optimized PDF" with nearly 60%. In the context of this survey "optimized PDF" simply means any PDF file that was specifically created for VDP.

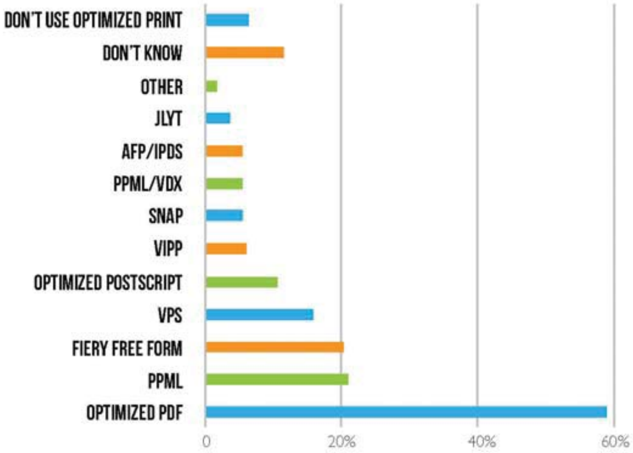


Fig. 1: Optimized print output language usage.

Users were asked to select the top two optimized print output formats used for variable data job production (Multiple responses permitted)
End-User Workflow Survey, InfoTrends 2010

For years many variable data print (VDP) vendors had insisted that you could only achieve high throughput on press by using specialist VDP print stream formats; the market no longer appears to disagree and is voting with its wallet. The survey was now conducted over three years ago, and anecdotal evidence is that the swing towards PDF has continued over that time.

PDF/VT doesn't appear in this chart because the standard was only published in 2010, the same year that the survey was published.

4.1 CHANGING DEMANDS ON VDP

Variable data is now printed at more print sites than ever before, driven by an overall growth in digital print, and by a transfer from printing customer mail in the data center to workflows that are more closely related to those found in the graphic arts.

Digital production presses and variable data print have developed greatly over the last decade or so. Presses are much faster than they used to be and often running at higher resolution, with more full color work. The extreme example of this is the new breed of ultra-high-speed continuous feed inkjet web presses, printing at over 500ft/min (150m/min) that emerged from 2009 onwards. The Hewlett-Packard T410 is an example of this class of press. The Screen Truepress Jet520 increases the data rate required by printing at 720dpi rather than the more common 600dpi. This means 120% as much raster data in each dimension (720/600), or nearly 50% more data required for each page.

Even in lower speed sectors such as high-volume cut-sheet, press speeds and duty cycles are increasing. The HP Indigo w7200 prints at 240 pages per minute (ppm) and the Indigo 10000 at 300ppm, for instance. Maximizing ROI on these presses requires that they be driven at or near full engine speed, for all of every shift, only stopping for scheduled maintenance.

On the up-side the computing power available for inclusion in a DFE has also been increasing, while its cost has dropped.

4.2 DEMANDS FOR RICHER DESIGNS

On balance it's probably now easier to RIP jobs fast enough to achieve full engine speed on a sheet-fed press than it used to be… or at least it is if you print the kind of simple VDP pages that were being processed a few years ago. A third trend that's occurred at the same time is that the complexity of VDP jobs has risen, increasing the demands on processing power in the DFE again.

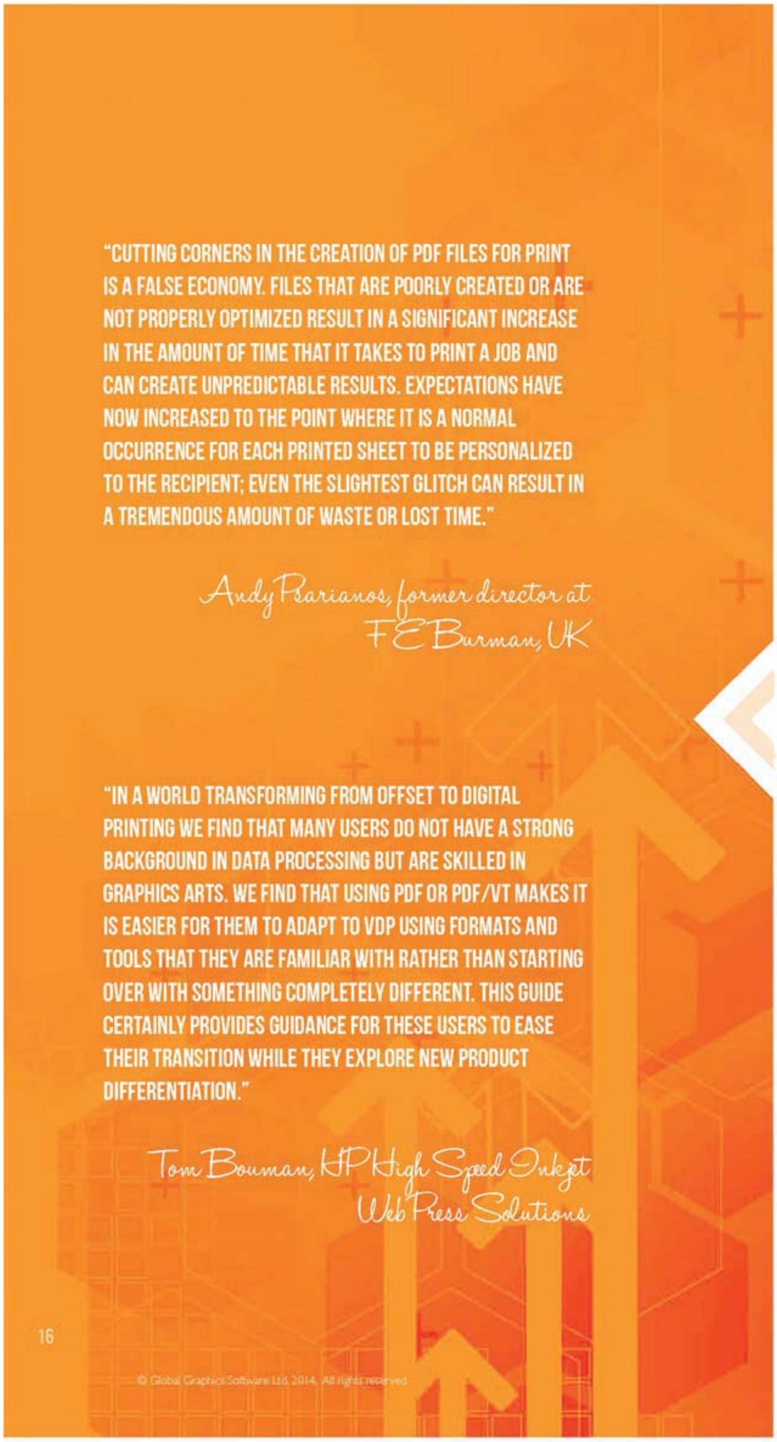
A successful direct marketing or transpromo campaign needs the printed product to be novel, attractive and compelling enough to persuade the recipient to read it before discarding it. The tools used by designers for creating general and publication print have become richer and more complex over time; designers for VDP pieces (quite naturally) want to take advantage of those tools, and there's often a demand for a common appearance between VDP pieces and, for instance advertising in magazines. This can lead to a tension between designers and the print production team over what features can be used in a VDP job while still achieving high enough performance in the DFE and on press to be commercially viable.

4.3 DRAWBACKS OF SPECIALIZED PDLS

Vendors have always tried to build solutions that are capable of the most efficient processing possible using the technology available at the time. Historically this led to the creation of a variety of specialist VDP page description languages (PDLs). By using something like the Print Production Markup Language (PPML) it was possible to reduce the amount of processing that the DFE had to do in order to achieve a given final appearance. The tools that create the PPML stream do some of the work for the DFE in identifying which parts of each page are used many times, so the DFE only needs to RIP each of those shared page elements once. It then RIPs all of the elements that were not shared. Finally the shared and variable elements for each page are stitched together (often using hardware assistance) and the page is printed.

That model may enable the highest possible throughput in the DFE and the press for relatively simple jobs, but it carries a number of hidden costs:

- There are many VDP-specific PDLs, some only supported by a single DFE or press vendor. A print site running presses from multiple suppliers may need to make files differently for each press, leading to higher costs for creation tools and training and a lack of flexibility in moving a job from press to press.
- Several proprietary VDP PDLs include assumptions that all DFEs that will process them include specialist hardware designed to stitch rasters post RIP. This makes it difficult to scale the use of exactly the same VDP PDL over a whole range of digital presses from light production to high-volume, again meaning that different PDLs are required for different printers and presses.
- Most VDP-specific PDLs were designed by a vendor who sells either a composition tool or a digital press with its associated DFE, so other aspects of the VDP production process are often not well served by the design. There's a lot more to workflow than making a VDP data stream in one place and printing it through a DFE and press at another, including viewing, proofing, approval, preflight etc.
- When most of the VDP-specific PDLs were first specified it was possible to use them to create pages as rich as those used in commercial and publication print at the time. Since then the use of



"CUTTING CORNERS IN THE CREATION OF PDF FILES FOR PRINT IS A FALSE ECONOMY. FILES THAT ARE POORLY CREATED OR ARE NOT PROPERLY OPTIMIZED RESULT IN A SIGNIFICANT INCREASE IN THE AMOUNT OF TIME THAT IT TAKES TO PRINT A JOB AND CAN CREATE UNPREDICTABLE RESULTS. EXPECTATIONS HAVE NOW INCREASED TO THE POINT WHERE IT IS A NORMAL OCCURRENCE FOR EACH PRINTED SHEET TO BE PERSONALIZED TO THE RECIPIENT; EVEN THE SLIGHTEST GLITCH CAN RESULT IN A TREMENDOUS AMOUNT OF WASTE OR LOST TIME."

Andy Psarianos, former director at
F E Burman, UK

"IN A WORLD TRANSFORMING FROM OFFSET TO DIGITAL PRINTING WE FIND THAT MANY USERS DO NOT HAVE A STRONG BACKGROUND IN DATA PROCESSING BUT ARE SKILLED IN GRAPHICS ARTS. WE FIND THAT USING PDF OR PDF/VT MAKES IT IS EASIER FOR THEM TO ADAPT TO VDP USING FORMATS AND TOOLS THAT THEY ARE FAMILIAR WITH RATHER THAN STARTING OVER WITH SOMETHING COMPLETELY DIFFERENT. THIS GUIDE CERTAINLY PROVIDES GUIDANCE FOR THESE USERS TO EASE THEIR TRANSITION WHILE THEY EXPLORE NEW PRODUCT DIFFERENTIATION."

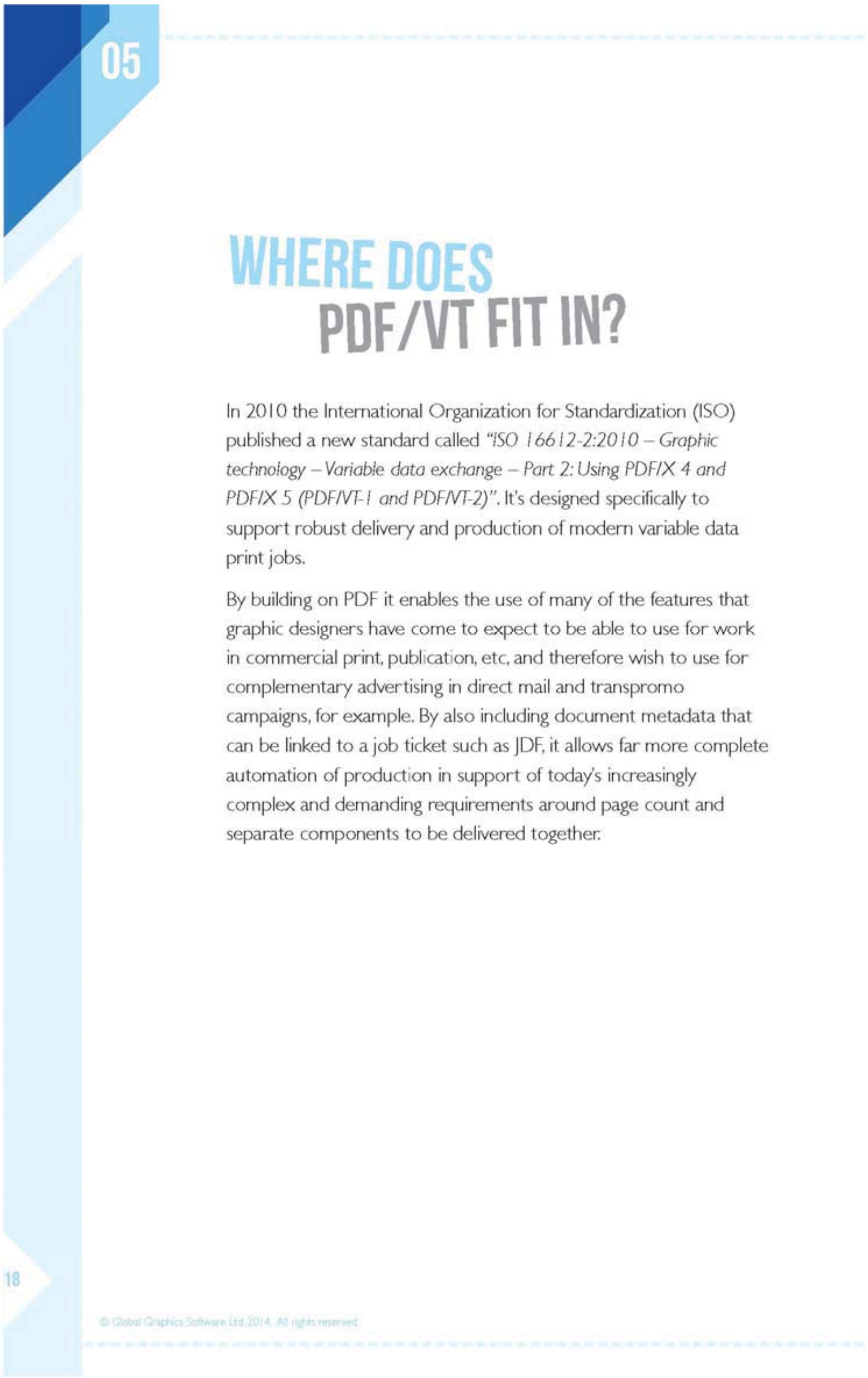
Tom Bouman, HP High Speed Inkjet
Web Press Solutions

live transparency in PDF has become commonplace. Many of the effects that can be delivered using current versions of PDF are either very difficult or impossible to reproduce efficiently in specialist VDP PDLs.

> PPML has now been updated to v3.0 to add limited support for live transparency, but most of the proprietary VDP PDLs have not. It's also remained true to its roots in constraining users to the graphical effects that can be processed most efficiently in today's DFEs. That's likely to be seen as overly restrictive as the next generations of DFEs for formats such as PDF/VT deliver higher performance without those limitations, allowing designers to match graphics used in VDP with those in commercial and publication collateral.

> Almost all long VDP jobs are created using specialist tools. But shorter VDP jobs created in-house by companies who have less frequent needs are often made with tools that were not designed to make VDP-specific PDLs. The print service provider (PSP) or corporate reproduction center (CRD) still needs to receive the documents in a stable, reliable format to be printed.

It's not all that surprising that a lot of companies creating VDP jobs, and print companies who print them have elected to use PDF instead of something more specialized to the task. The ability to explain to all customers what they need to submit, to send the same file to (almost) all DFEs, to view the final file virtually anywhere, and to create files as rich as the customer demands all go at least some way to balancing out the potential for a drop in performance in the DFE, especially as that drop will be very small if the recommendations on this guide are followed.

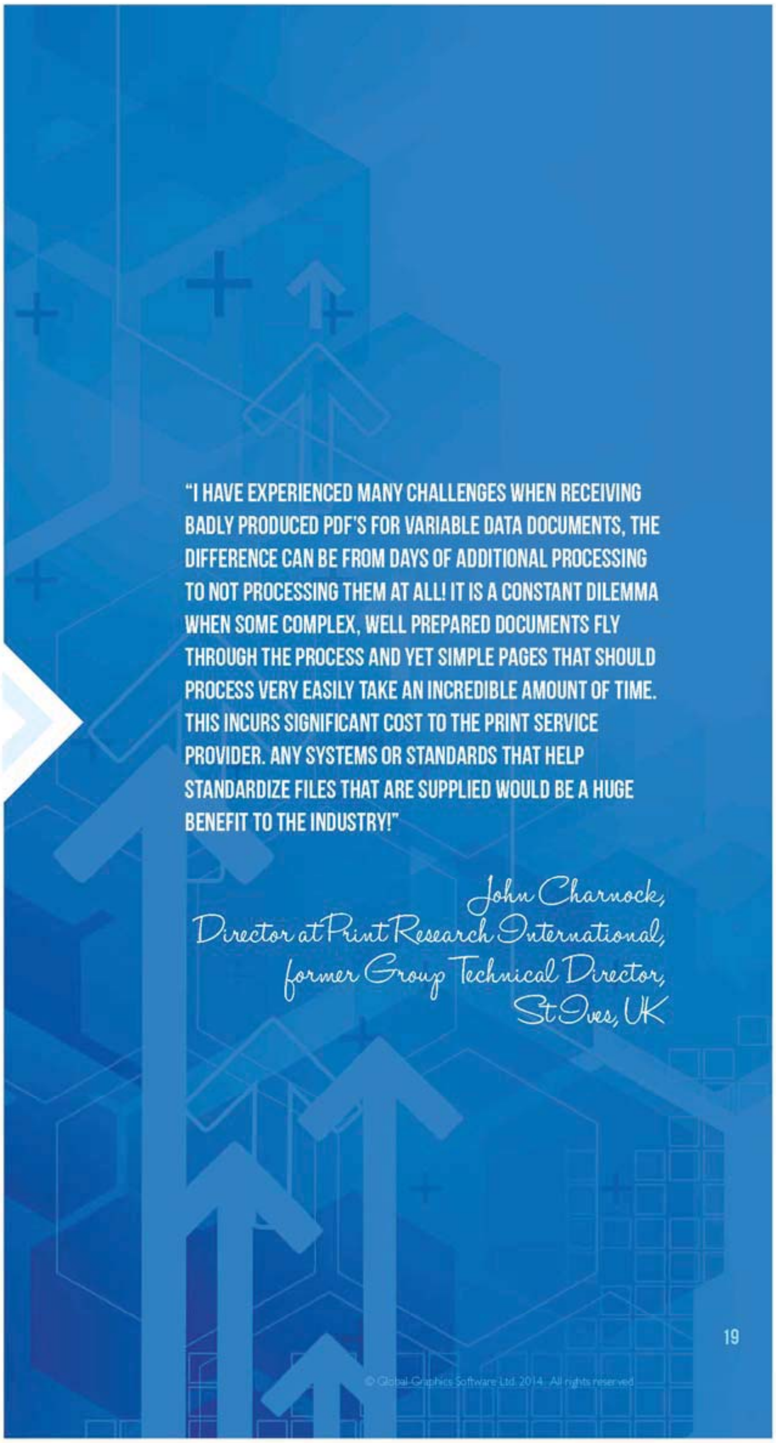


05

WHERE DOES PDF/VT FIT IN?

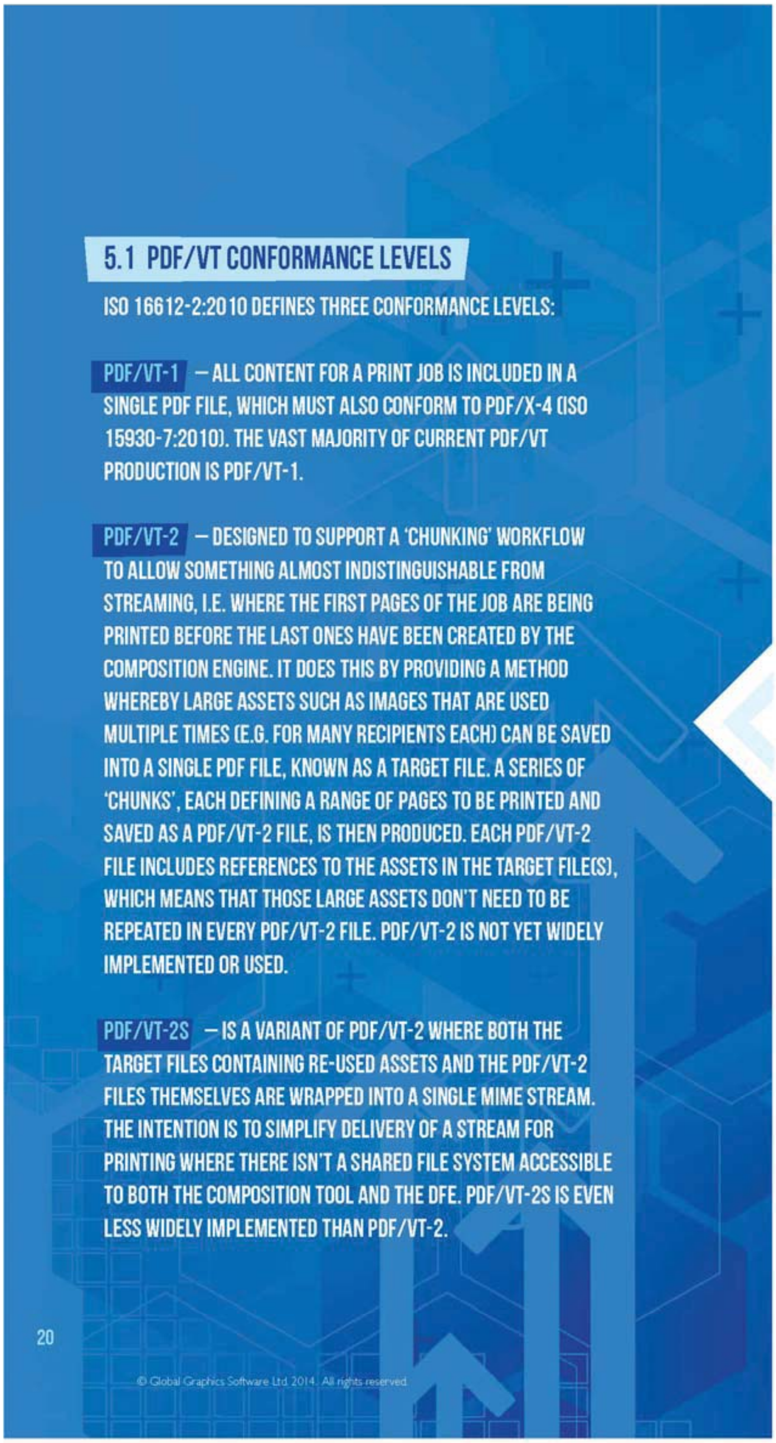
In 2010 the International Organization for Standardization (ISO) published a new standard called *"ISO 16612-2:2010 – Graphic technology – Variable data exchange – Part 2: Using PDF/X 4 and PDF/X 5 (PDF/VT-1 and PDF/VT-2)"*. It's designed specifically to support robust delivery and production of modern variable data print jobs.

By building on PDF it enables the use of many of the features that graphic designers have come to expect to be able to use for work in commercial print, publication, etc, and therefore wish to use for complementary advertising in direct mail and transpromo campaigns, for example. By also including document metadata that can be linked to a job ticket such as JDF, it allows far more complete automation of production in support of today's increasingly complex and demanding requirements around page count and separate components to be delivered together.



"I HAVE EXPERIENCED MANY CHALLENGES WHEN RECEIVING BADLY PRODUCED PDF'S FOR VARIABLE DATA DOCUMENTS, THE DIFFERENCE CAN BE FROM DAYS OF ADDITIONAL PROCESSING TO NOT PROCESSING THEM AT ALL! IT IS A CONSTANT DILEMMA WHEN SOME COMPLEX, WELL PREPARED DOCUMENTS FLY THROUGH THE PROCESS AND YET SIMPLE PAGES THAT SHOULD PROCESS VERY EASILY TAKE AN INCREDIBLE AMOUNT OF TIME. THIS INCURS SIGNIFICANT COST TO THE PRINT SERVICE PROVIDER. ANY SYSTEMS OR STANDARDS THAT HELP STANDARDIZE FILES THAT ARE SUPPLIED WOULD BE A HUGE BENEFIT TO THE INDUSTRY!"

John Charnock,
Director at Print Research International,
former Group Technical Director,
St Ives, UK



5.1 PDF/VT CONFORMANCE LEVELS

ISO 16612-2:2010 DEFINES THREE CONFORMANCE LEVELS:

PDF/VT-1 – ALL CONTENT FOR A PRINT JOB IS INCLUDED IN A SINGLE PDF FILE, WHICH MUST ALSO CONFORM TO PDF/X-4 (ISO 15930-7:2010). THE VAST MAJORITY OF CURRENT PDF/VT PRODUCTION IS PDF/VT-1.

PDF/VT-2 – DESIGNED TO SUPPORT A 'CHUNKING' WORKFLOW TO ALLOW SOMETHING ALMOST INDISTINGUISHABLE FROM STREAMING, I.E. WHERE THE FIRST PAGES OF THE JOB ARE BEING PRINTED BEFORE THE LAST ONES HAVE BEEN CREATED BY THE COMPOSITION ENGINE. IT DOES THIS BY PROVIDING A METHOD WHEREBY LARGE ASSETS SUCH AS IMAGES THAT ARE USED MULTIPLE TIMES (E.G. FOR MANY RECIPIENTS EACH) CAN BE SAVED INTO A SINGLE PDF FILE, KNOWN AS A TARGET FILE. A SERIES OF 'CHUNKS', EACH DEFINING A RANGE OF PAGES TO BE PRINTED AND SAVED AS A PDF/VT-2 FILE, IS THEN PRODUCED. EACH PDF/VT-2 FILE INCLUDES REFERENCES TO THE ASSETS IN THE TARGET FILE(S), WHICH MEANS THAT THOSE LARGE ASSETS DON'T NEED TO BE REPEATED IN EVERY PDF/VT-2 FILE. PDF/VT-2 IS NOT YET WIDELY IMPLEMENTED OR USED.

PDF/VT-2S – IS A VARIANT OF PDF/VT-2 WHERE BOTH THE TARGET FILES CONTAINING RE-USED ASSETS AND THE PDF/VT-2 FILES THEMSELVES ARE WRAPPED INTO A SINGLE MIME STREAM. THE INTENTION IS TO SIMPLIFY DELIVERY OF A STREAM FOR PRINTING WHERE THERE ISN'T A SHARED FILE SYSTEM ACCESSIBLE TO BOTH THE COMPOSITION TOOL AND THE DFE. PDF/VT-2S IS EVEN LESS WIDELY IMPLEMENTED THAN PDF/VT-2.

The PDF/VT standard mandates that certain items are included in a file when it's created which are extremely useful in ensuring that best practice is followed. As an example, it requires that all fonts needed to RIP the job are embedded within the file. In a sense it relieves the graphic designer and composition tool operator of the need to consider some of these items when they make a file; just select "PDF/VT" in the menu when generating the file for print and it will be done for you.

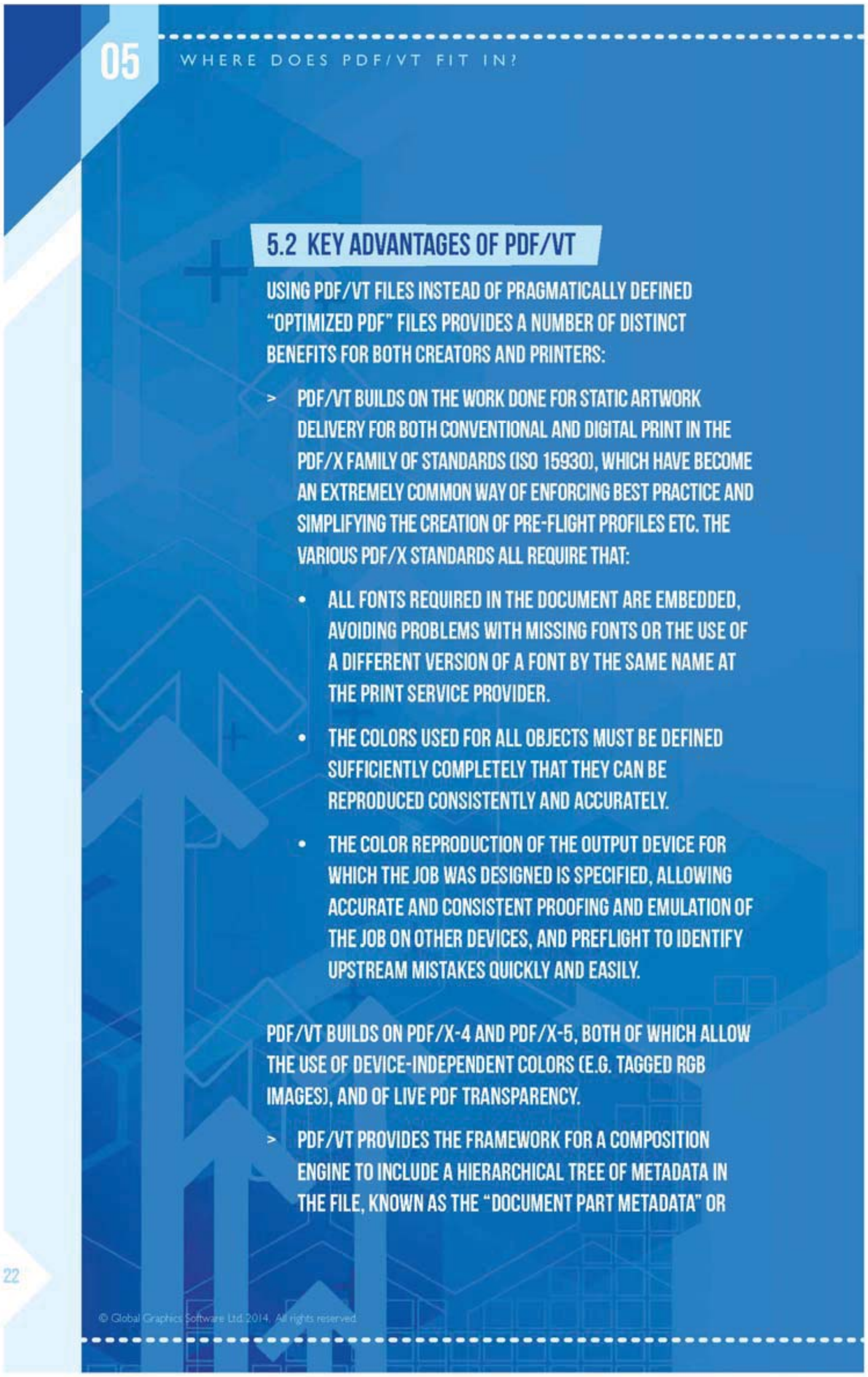
But the PDF/VT standard concentrates on providing support for predictable and repeatable output and for automation; it does not focus on how the desired elements should be written into that file in order to maximize the efficiency of processing.

So using PDF/VT is a very good way of improving the document delivery workflow in many ways, and is definitely recommended.

But it's not the whole story. There are many things that users can do to optimize processing of those jobs as well, and to help avoid last-minute problems. Those are the subject of this guide, and most are equally applicable to both PDF/VT and 'baseline' PDF.

"FOR DIGITAL PRINTING PDF/X-4 (WHICH IS THE BASE OF PDF/VT) IS VERY IMPORTANT. OLDER STANDARDS LIKE PDF/X-1A AND PDF/X-3 REQUIRE TRANSPARENCY FLATTENING WHICH CAN EASILY CONVERT A FEW DOZEN OBJECTS INTO THOUSANDS (OR MORE) OBJECTS WHICH SLOWS DOWN EVEN HIGH SPEED RIPS SIGNIFICANTLY AND PREVENTS HIGH PRODUCTION DIGITAL PRINTERS FROM RUNNING AT ENGINE SPEED."

Stephan Jaeggi, PrePress-Consulting
Switzerland



5.2 KEY ADVANTAGES OF PDF/VT

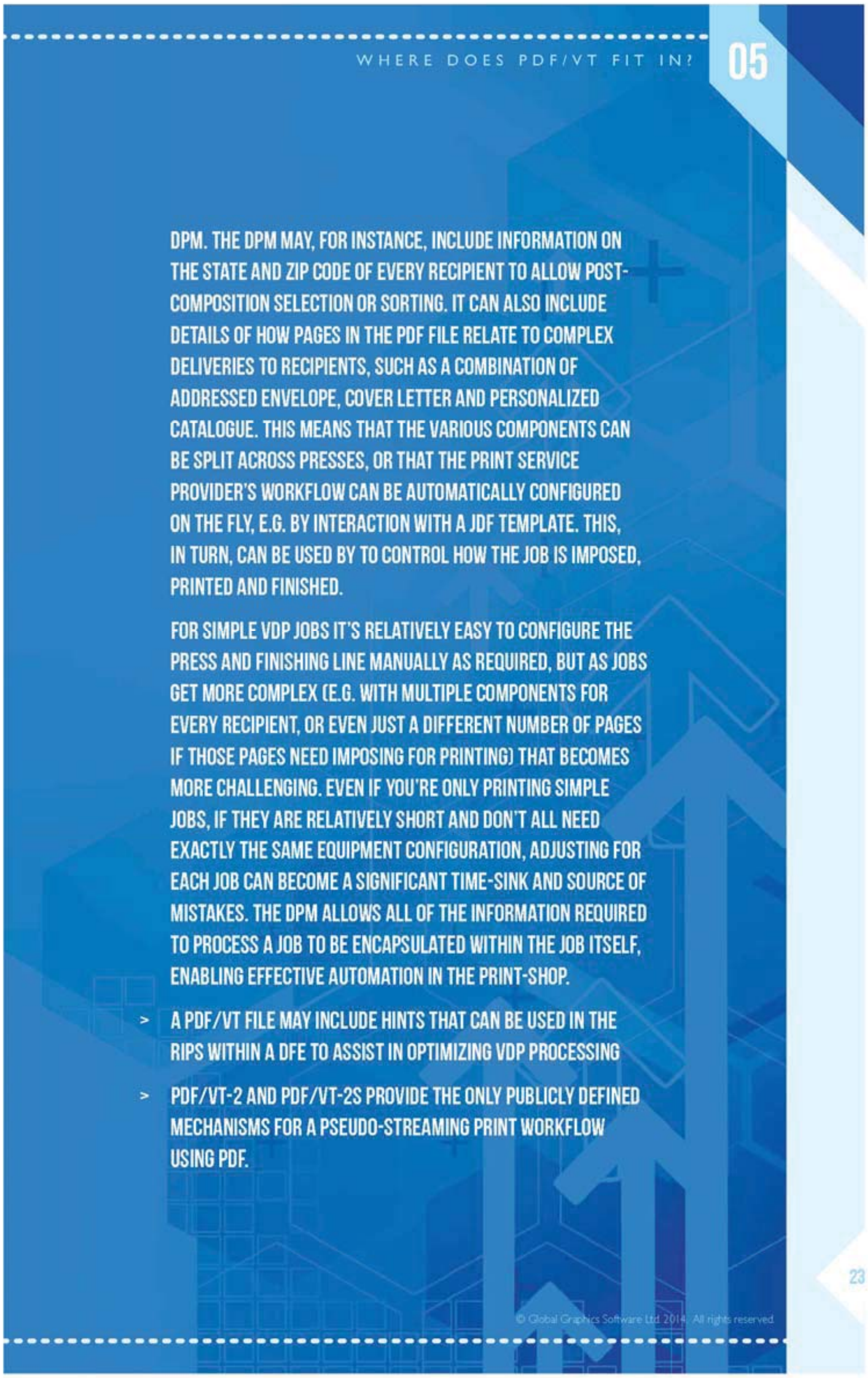
USING PDF/VT FILES INSTEAD OF PRAGMATICALLY DEFINED "OPTIMIZED PDF" FILES PROVIDES A NUMBER OF DISTINCT BENEFITS FOR BOTH CREATORS AND PRINTERS:

> PDF/VT BUILDS ON THE WORK DONE FOR STATIC ARTWORK DELIVERY FOR BOTH CONVENTIONAL AND DIGITAL PRINT IN THE PDF/X FAMILY OF STANDARDS (ISO 15930), WHICH HAVE BECOME AN EXTREMELY COMMON WAY OF ENFORCING BEST PRACTICE AND SIMPLIFYING THE CREATION OF PRE-FLIGHT PROFILES ETC. THE VARIOUS PDF/X STANDARDS ALL REQUIRE THAT:

- ALL FONTS REQUIRED IN THE DOCUMENT ARE EMBEDDED, AVOIDING PROBLEMS WITH MISSING FONTS OR THE USE OF A DIFFERENT VERSION OF A FONT BY THE SAME NAME AT THE PRINT SERVICE PROVIDER.
- THE COLORS USED FOR ALL OBJECTS MUST BE DEFINED SUFFICIENTLY COMPLETELY THAT THEY CAN BE REPRODUCED CONSISTENTLY AND ACCURATELY.
- THE COLOR REPRODUCTION OF THE OUTPUT DEVICE FOR WHICH THE JOB WAS DESIGNED IS SPECIFIED, ALLOWING ACCURATE AND CONSISTENT PROOFING AND EMULATION OF THE JOB ON OTHER DEVICES, AND PREFLIGHT TO IDENTIFY UPSTREAM MISTAKES QUICKLY AND EASILY.

PDF/VT BUILDS ON PDF/X-4 AND PDF/X-5, BOTH OF WHICH ALLOW THE USE OF DEVICE-INDEPENDENT COLORS (E.G. TAGGED RGB IMAGES), AND OF LIVE PDF TRANSPARENCY.

> PDF/VT PROVIDES THE FRAMEWORK FOR A COMPOSITION ENGINE TO INCLUDE A HIERARCHICAL TREE OF METADATA IN THE FILE, KNOWN AS THE "DOCUMENT PART METADATA" OR



DPM. THE DPM MAY, FOR INSTANCE, INCLUDE INFORMATION ON THE STATE AND ZIP CODE OF EVERY RECIPIENT TO ALLOW POST-COMPOSITION SELECTION OR SORTING. IT CAN ALSO INCLUDE DETAILS OF HOW PAGES IN THE PDF FILE RELATE TO COMPLEX DELIVERIES TO RECIPIENTS, SUCH AS A COMBINATION OF ADDRESSED ENVELOPE, COVER LETTER AND PERSONALIZED CATALOGUE. THIS MEANS THAT THE VARIOUS COMPONENTS CAN BE SPLIT ACROSS PRESSES, OR THAT THE PRINT SERVICE PROVIDER'S WORKFLOW CAN BE AUTOMATICALLY CONFIGURED ON THE FLY, E.G. BY INTERACTION WITH A JDF TEMPLATE. THIS, IN TURN, CAN BE USED BY TO CONTROL HOW THE JOB IS IMPOSED, PRINTED AND FINISHED.

FOR SIMPLE VDP JOBS IT'S RELATIVELY EASY TO CONFIGURE THE PRESS AND FINISHING LINE MANUALLY AS REQUIRED, BUT AS JOBS GET MORE COMPLEX (E.G. WITH MULTIPLE COMPONENTS FOR EVERY RECIPIENT, OR EVEN JUST A DIFFERENT NUMBER OF PAGES IF THOSE PAGES NEED IMPOSING FOR PRINTING) THAT BECOMES MORE CHALLENGING. EVEN IF YOU'RE ONLY PRINTING SIMPLE JOBS, IF THEY ARE RELATIVELY SHORT AND DON'T ALL NEED EXACTLY THE SAME EQUIPMENT CONFIGURATION, ADJUSTING FOR EACH JOB CAN BECOME A SIGNIFICANT TIME-SINK AND SOURCE OF MISTAKES. THE DPM ALLOWS ALL OF THE INFORMATION REQUIRED TO PROCESS A JOB TO BE ENCAPSULATED WITHIN THE JOB ITSELF, ENABLING EFFECTIVE AUTOMATION IN THE PRINT-SHOP.

> A PDF/VT FILE MAY INCLUDE HINTS THAT CAN BE USED IN THE RIPS WITHIN A DFE TO ASSIST IN OPTIMIZING VDP PROCESSING

> PDF/VT-2 AND PDF/VT-2S PROVIDE THE ONLY PUBLICLY DEFINED MECHANISMS FOR A PSEUDO-STREAMING PRINT WORKFLOW USING PDF.

06

VARIATIONS IN COMPOSITION WORKFLOWS

Over the years a very wide variety of different approaches to composition for VDP have emerged. To over-simplify the market slightly, most of these can be characterized as taking various positions along a scale between two points:

'SIMPLE': Complete 'backgrounds' for variable data jobs are created by a graphic designer. The composition tool can import those backgrounds as assets which can only be placed and positioned as atomic units. Relatively simple variable text and graphics can then be placed, usually over the top of the background.

'COMPLEX': A wide variety of assets can be imported into the composition tool as individual images, logos and graphics. Both variable text and other data and a variety of rules can be added which select which assets should be placed for each recipient of the printed piece and where those should be positioned. In some cases variable data can be used to construct graphics on the fly, such as pie charts for financial reports, or personalized images, such as those created by DirectSmile.

Many tools can be used in a way that allows the operator to select where their particular usage falls on this spectrum. As an example, even when a tool capable of a 'complex' design process way is used, it's often possible to place an asset that happens to be a full-page PDF file which already contains multiple and potentially complex graphics.

This variation in tools and in how they can be used implies an equal variation in how all the people involved in creating a job can affect the efficiency of the resulting PDF or PDF/VT file, and therefore where the responsibility for doing so will fall.

If a composition tool is used at the 'simple' end of the spectrum then all of the choices about the use of transparency, image resolution, font embedding etc are likely to have been made by the graphic designer when creating the 'background' assets. The composition tool will quite often include the PDF objects from the asset into the PDF file for printing as-is, without making any changes other than those required to reference them from the composited pages. In this scenario the graphic designer is largely responsible for the efficiency of the result.

At the other end of the scale, a composition tool may be used in 'complex' mode, where all assets are supplied as single images and relatively simple graphics and the rules defined in the composition engine trigger the majority of the way in which the PDF print file is constructed. Responsibility for how efficiently the job prints in this case is shared between the designer, the developers at the composition tool vendor and the operator of that tool, depending on the richness of the rule set that can be used. If the rule set allows the operator to make significant decisions around the use of transparency, for instance, those decisions must be made wisely. On the other hand, the tool itself will usually make decisions about if, and how, fonts should be embedded. If image down-sampling is available it may be configurable by the user, or be applied automatically by the tool.

Composition tool vendors may also provide some relatively simple pre-flight feedback to identify use of assets or decisions by the operator that might reduce efficiency at the print site.

07

HIGH LEVEL VIEW OF VDP OPTIMIZATIONS: RIP ONCE, USE MANY TIMES

A very short run of a commercial or publication job on a digital press tends to mean that you're probably still producing a few dozen copies. In other words, each page is processed once in the DFE for the press (color managed, RIPed, maybe trapped, screened etc), and then sent multiple times to the press. The DFE doesn't need to process pages at the same speed that the press engine can print them. But if you're printing a variable data job it's likely that many pages will be unique; most pages will be at least slightly different to every other page. Obviously this is not a universal rule; if you're printing invoices, for example, it's common for the back of every sheet to be the same as the back of every other … but even then there may be an invoice number or date added onto the back of the sheet.

Building a DFE to be able to process whole pages as fast as the engine could consume them is relatively expensive, so the DFEs for many digital production presses include optimizations designed specifically to handle VDP jobs.

When a VDP piece is designed a variety of assets of various forms are collected together. Some assets are intended to be used multiple times, while others are associated with a single recipient or personalization. They may be images, graphics (e.g. maps), static text blocks, variable text and even variable images and graphics.

All of the assets are placed and positioned according to a set of rules. Those rules might be as simple as a mail merge in Microsoft Word, where placeholders are included in a template for the document, and then replaced with text from a separate data file. In more sophisticated environments additional information from a database about each recipient is used to select from the assets available. Thus 'platinum' members of an organization may see one version of an asset, while 'gold' or 'basic' members see different ones.

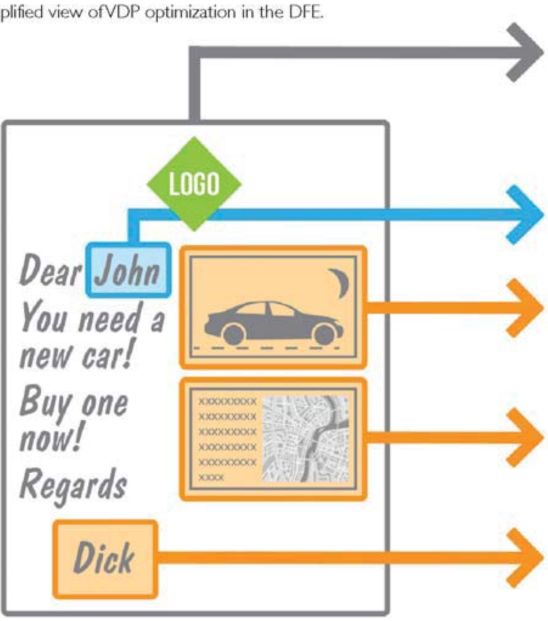
A classic direct marketing example is a mailer sent out to people who have previously bought a particular make of car a couple of years after that purchase to invite them to come in and view this year's model. Each piece might include a photograph of a car of the same class as the one they purchased and perhaps in the same color. Thus, if they had bought a sedan they'd see an image of this year's sedan, if they bought a sports car they'd see a sports car. In addition there might be a map to the dealer that they bought from last time, the name and contact details for an appropriate sales representative, etc.

All of the assets required to reproduce the pages are then included in the PDF file and sent to be printed. The PDF can be viewed in any PDF reader and would display as a series of fully laid out pages. It could be processed through a DFE in that way as well … but often not at high enough speed to keep the press itself busy.

"WITH MORE AND MORE CREATIVE AGENCIES EMBRACING THE POSSIBILITIES OF VARIABLE COLOUR, ISSUES WITH BADLY PREPARED VARIABLE DATA IS SOMETHING OUR SUPPORT TEAM DEAL WITH ON A REGULAR BASIS. THE USE OF COLOUR FOR VARIABLE DATA PRINTING IS ONLY GOING TO INCREASE AS ARE THE SPEEDS OF INKJET PRESSES SO THE TIMING IS PERFECT FOR A SIMPLE GUIDE TO FILE PREPARATION. BY FOLLOWING A FEW BASIC RULES YOU CAN TRANSFORM THE 'PRINTABILITY' OF A JOB WITHOUT ANY NEED TO COMPROMISE ON DESIGN INTEGRITY."

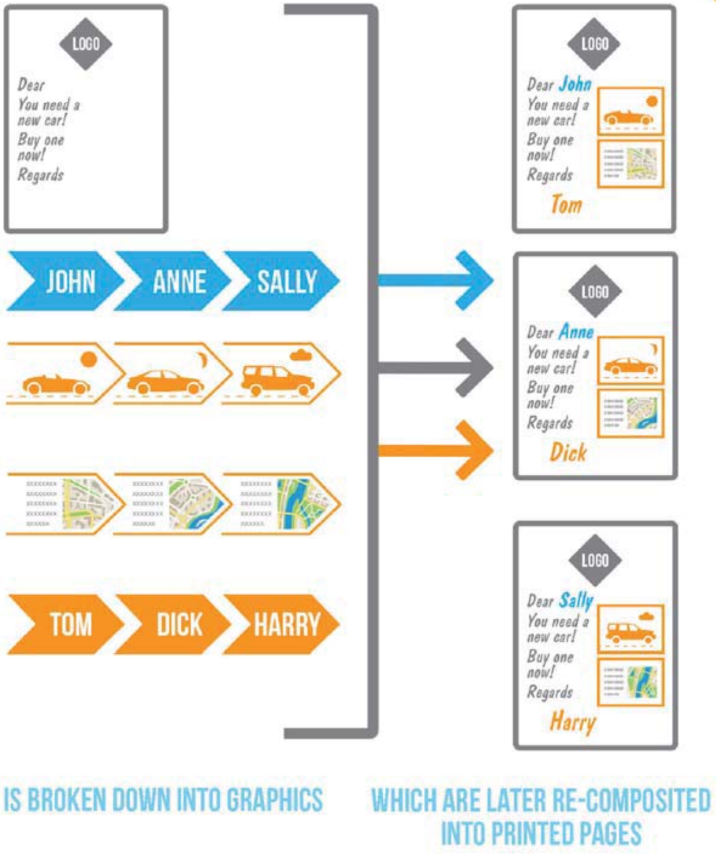
Tim Taylor, VP Solutions &
Technology Screen Europe

Fig. 2: a simplified view ofVDP optimization in the DFE.



THE PDF FILE PAGE

The optimization process in a DFE is usually more or less the opposite of how the composition engines built the print stream in the first place. The PDF file is examined to identify graphics that are used as a group multiple times. Those are then processed separately and stored, along with data recording where they were seen in the job. Those elements of each page that are only used once are also processed. Processing here typically means applying all color



management and rendering to a raster format at the same resolution, and using the same colorants, as the digital press.

Finally, the various components that make up each page of the job are re-composited. While this may seem a very complex method it means that many of the original assets from the design phase are only processed once in the DFE.

XOBJECTS ARE EFFECTIVELY GRAPHICAL ASSETS DEFINED WITHIN THE PDF FILE ITSELF. EACH ONE HOLDS AN IMAGE, OR A COLLECTION OF GRAPHICS SUCH AS TEXT, RULES AND COLOR FILLS IN A WAY THAT MEANS THEY CAN BE RE-USED MANY TIMES IN THE SAME JOB WITHOUT HAVING TO INCLUDE THE DATA MULTIPLE TIMES. XOBJECTS CAN BE NESTED INTO COMPLEX COMBINATIONS OF GRAPHICS AND/OR IMAGES TOGETHER.

Different technologies from various vendors handle each step of this process in different ways. Some, for example identify re-used elements by looking at the number of references to PDF constructs called XObjects, while others review all graphics and identify sequences that are repeated irrespective of how they are structured into objects within the file.

Once re-used elements are identified some systems coalesce them together by determining which collections of elements are used together on multiple pages with consistent positioning relative to each other. This minimizes the number of components required to construct every final page in the job. Achieving this coalescing automatically, flexibly and intelligently has a huge impact on the overall throughput of the DFE and is a key distinguishing factor between RIPs and DFEs from different vendors.

> **"UNPREDICTABLE DELAYS CAN WREAK HAVOC WITH LOW MARGIN/HIGH VOLUME JOBS OFTEN FOUND IN BUSINESS/TRANSACTIONAL APPLICATIONS. IN SOME CASES EXTRA STAFF MUST BE MAINTAINED TO DIAGNOSE AND PROVIDE WORKAROUNDS FOR PROBLEM JOBS IN THE VDP ARENA."**
>
> *Mike Rodriguez independent color consultant and former director at RR Donnelley, USA*

08

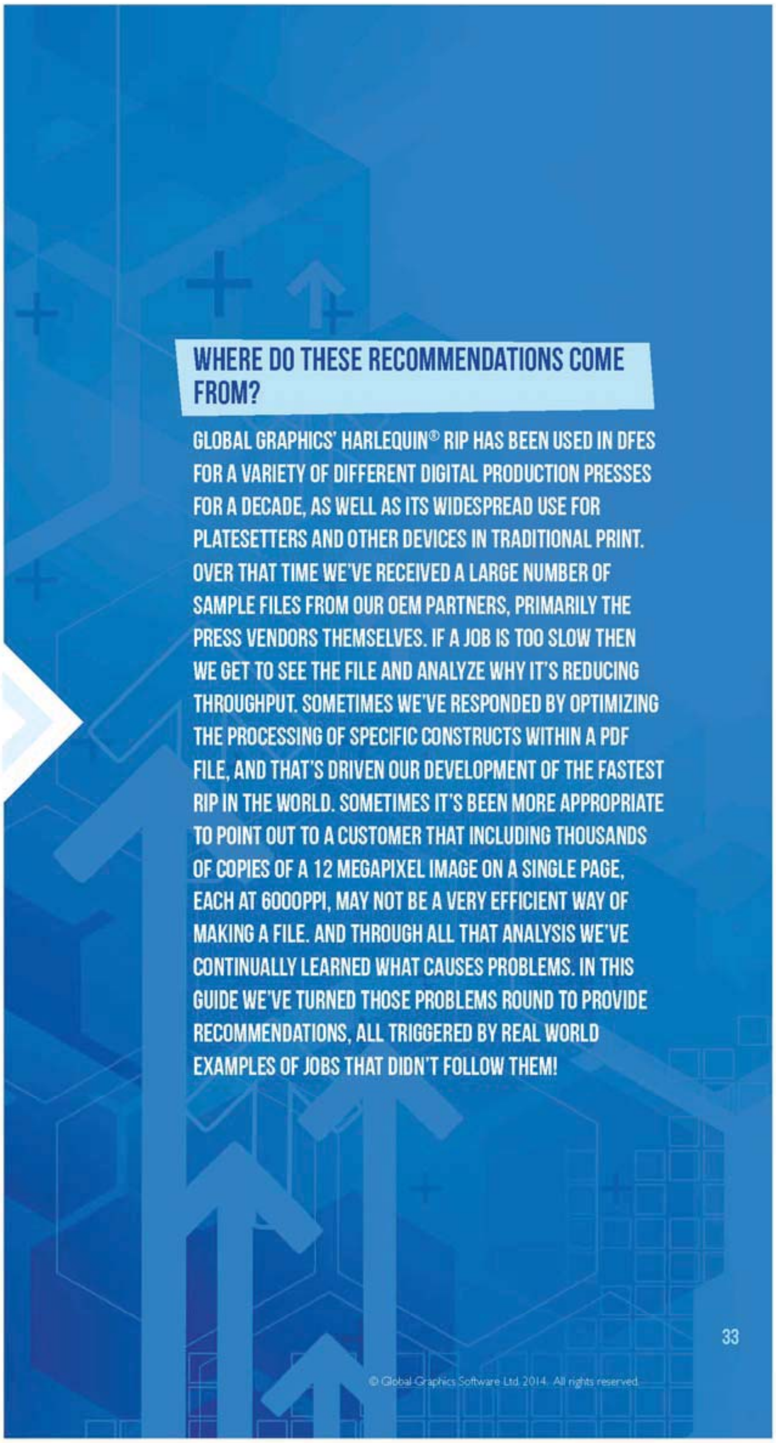
MAKING EFFICIENT PDF FILES

This section sets out a number of guidelines for avoiding tripping up the print production workflow with your PDF files forVDP. At the highest level almost all of them boil down to a very simple maxim: don't ask the print workflow to do more work than necessary if that doesn't change the look of the printed result.

In every print workflow there is always one rule that overrides virtually everything else: the printed result must be what the person signing the check wanted and expected. This guide is not intended to restrict the ability of marketing departments and graphic designers to achieve the desired visual appearance of printed work. It provides guidance on easing the path to the most efficient production of that design … whatever that desired result might be.

There are often multiple ways of achieving the same visual appearance which can vary significantly in the amount of processing required to print them. Sometimes the most efficient method for the print company requires a little more work for the origination company, and sometimes there's a win-win where improved print performance can be gained for just a few seconds of thought upstream.

The effect of much of the advice below, such as using images at an optimal resolution or discarding cropped image pixels, will vary significantly depending on how the graphics in question are used in the job. Optimizing an image that is used in exactly the same way on the output for every recipient of the job will have a very minor impact, because a well-designed DFE will only process that image a few times (possibly only once) and re-use the results multiple times. On the other hand, optimizing images that are personal to every recipient (e.g. images custom-built to include the recipient's name) can



WHERE DO THESE RECOMMENDATIONS COME FROM?

GLOBAL GRAPHICS' HARLEQUIN® RIP HAS BEEN USED IN DFES FOR A VARIETY OF DIFFERENT DIGITAL PRODUCTION PRESSES FOR A DECADE, AS WELL AS ITS WIDESPREAD USE FOR PLATESETTERS AND OTHER DEVICES IN TRADITIONAL PRINT. OVER THAT TIME WE'VE RECEIVED A LARGE NUMBER OF SAMPLE FILES FROM OUR OEM PARTNERS, PRIMARILY THE PRESS VENDORS THEMSELVES. IF A JOB IS TOO SLOW THEN WE GET TO SEE THE FILE AND ANALYZE WHY IT'S REDUCING THROUGHPUT. SOMETIMES WE'VE RESPONDED BY OPTIMIZING THE PROCESSING OF SPECIFIC CONSTRUCTS WITHIN A PDF FILE, AND THAT'S DRIVEN OUR DEVELOPMENT OF THE FASTEST RIP IN THE WORLD. SOMETIMES IT'S BEEN MORE APPROPRIATE TO POINT OUT TO A CUSTOMER THAT INCLUDING THOUSANDS OF COPIES OF A 12 MEGAPIXEL IMAGE ON A SINGLE PAGE, EACH AT 6000PPI, MAY NOT BE A VERY EFFICIENT WAY OF MAKING A FILE. AND THROUGH ALL THAT ANALYSIS WE'VE CONTINUALLY LEARNED WHAT CAUSES PROBLEMS. IN THIS GUIDE WE'VE TURNED THOSE PROBLEMS ROUND TO PROVIDE RECOMMENDATIONS, ALL TRIGGERED BY REAL WORLD EXAMPLES OF JOBS THAT DIDN'T FOLLOW THEM!

have a huge effect because those images must be processed many times, once for every single recipient. Graphics that are used for some subset of the recipients, usually based on some metadata about the recipient, fall somewhere in between. If you only have the time to focus on parts of your workflow you should concentrate on the graphics that are individual to each recipient.

Most of these recommendations are relevant to the designers and composition operators in the trenches. A few are so deeply into the technical details of constructing a PDF file that they can only really be addressed by the developers who create and maintain the VDP workflow software that we all depend on. Those few have been split out to a separate section at the end.

8.1 USE PDF/VT

Section 5 set out some of the advantages of using PDF/VT instead of baseline PDF, but it's worth reiterating as a specific recommendation: use PDF/VT when you can.

8.2 OPTIMIZING IMAGES

As a general rule images tend to take longer than vector graphics and text to process in a DFE. A photographic image will often use quite a large number of different colors, each of which must be appropriately color managed. In addition there is simply more data involved which must often be copied between memory locations, and the difference between the effective resolution of the source image and the resolution of the output device must be resolved.

These operations only take a few milliseconds individually, but multiplied over all the images in a job they can amount to a significant total time.

At the same time images are commonly re-used within a VDP job; they may form part of a static page background, or a small number of images may be selected from, each being used for a proportion of the recipients (like the car images in the example in section 7, for instance). Thus being able to process each of a relatively small number of images only once, and then re-use the result many times can significantly increase the throughput of the DFE.

It's worth noting that many of these recommendations around image handling will also make a PDF file more appropriate for multi-channel delivery, e.g. by the web, email or to a mobile device because they will reduce the file size and allow a more resource constrained viewer to display them correctly.

8.2.1 SET PHOTOGRAPHIC IMAGE RESOLUTIONS APPROPRIATELY

There's a general rule of thumb in conventional print that you shouldn't place photographic images with an effective resolution greater than double the halftone screen frequency that you're using, because you won't gain any quality from going higher. So if you're screening at 150lpi, for instance, images should normally be included at, or just under 300ppi (pixels per inch).

The most appropriate image resolution varies somewhat for each digital press, depending on the printing heads, media and screening used, but aiming at around 300ppi is still a pretty good target for most. Using an effective image resolution higher than the output resolution of the press is virtually never productive.

The image content can also affect this slightly; a soft and dreamy image can often be placed at a significantly lower resolution, while one with high-contrast fine detail may benefit from a slightly higher one. To play safe in an automated workflow you may choose to select a resolution that is enough to maximize quality for the sharpest and most detailed images, say 350ppi or, for best results, ask your digital press vendor what they recommend.

As you can see from the side-bar it can be very easy to use an image at several times the required resolution. In the example the image is at 700ppi on the page, at least double what is required. That doubling applies in both the height and width of the image, so there are actually four times as many pixels as necessary, which can significantly impact performance in the DFE. Just imagine what would happen if the same image file had been placed at only 2 x 3 inches (5 x 7.5cm), there would then be 16 (4x4) times as much data as required.

A variety of tools are available for optimizing image resolution, and some composition tools can also do this automatically.

Note that this section applies only to photographic images (where each pixel may represent one of a number of tone values for each colorant) including both color and grayscale. Copy-dot scans, screen grabs and other synthetic images usually benefit from higher effective resolutions, with the optimal value normally being at the same resolution as the press itself, but watch out for moiré between the original image resolution and the press resolution if you don't match them exactly.

8.2.2 DISCARD CROPPED PIXELS FROM IMAGES

If an image is heavily cropped then the portions outside the cropped area should be completely discarded rather than simply hidden using a clipping path. Even though the clipped out pixels won't typically be color managed etc, they will typically still need to be read from the PDF file in order to find the pixels that are actually required.

EFFECTIVE IMAGE RESOLUTION

WHEN AN IMAGE IS PLACED ONTO A PAGE THE ORIGINAL RESOLUTION OF THAT IMAGE IS LARGELY IRRELEVANT; WHAT MATTERS IS HOW MANY PIXELS THERE ARE PER INCH ON THE FINAL PRINTED PAGE. AS AN EXAMPLE, IF YOU HAVE A PHOTOGRAPH FROM A 12MP DIGITAL SLR IT'LL PROBABLY BE APPROXIMATELY 2800 PIXELS BY 4200 PIXELS. IF THAT'S PLACED ON THE PAGE AS 4 INCHES BY 6 INCHES (10 X 15CM) THEN THE EFFECTIVE RESOLUTION IS ABOUT 700PPI (2800/4).

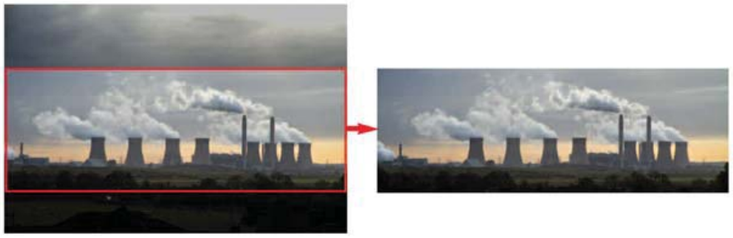


Fig. 3: Discard cropped pixels from images.

Cropping images can sometimes be efficiently combined with a resolution reduction step.

8.2.3 OPTIMIZING PERSONALIZED IMAGES

Some asset creation or composition tools, such as DirectSmile, can create images that are personalized for each recipient of a VDP piece. In most cases the proportion of the image that carries the personalization is quite small. It is often more efficient for the whole image, without personalization, to be included once for all recipients, with a smaller image (or images) overlaid in the correct position to carry the personalized area. This means that the un-personalized image can be treated as static data and processed once even though it appears on many pages. The personalized image(s) will be treated as variable data and processed for every recipient … but being much smaller that processing won't take as long.

Of course, it's vital that the small, personalized, image(s) are exactly aligned with the whole background image and set to use exactly the same halftones to avoid any artifacts along their edges.

8.2.4 AVOID IMAGE INTERPOLATION

The PDF specification includes a flag that can be included in an image to instruct the DFE to interpolate or up-sample the image. Interpolation is a relatively slow process and should be avoided if possible. If a photograph is used at such a size that it does not achieve the minimum image resolution appropriate for your press should be up-sampled during or before the creation of the PDF. Ideally you may wish to consider the use of a different image, or to crop it less tightly to ensure that you achieve a high quality print. If neither can be done the image should be included as-is, without requesting interpolation; the image quality is unlikely to be noticeably different from an interpolated one.

8.3 OPTIMIZING TRANSPARENCY

The very rich and flexible support for live transparency in PDF is an incredibly useful aspect of the format, and is one of the key reasons for selecting PDF over other page description languages for production print. On the other hand compositing transparent regions in a PDF file is much more processor intensive than handling opaque areas of a page.

As an example, consider two overlapping RGB images, both tagged with an ICC profile for ECI RGB in a PDF file.

When outputting to a digital press printing in CMYK with no live transparency involved the color of each pixel in each image must be transformed into tone values for CMYK, usually using ICC profiles. In most DFEs the results of the calculation for each set of RGB values from the image will be cached and re-used when another pixel using exactly the same RGB values is processed. There's a reasonable amount of processing involved, but nothing too heavyweight.

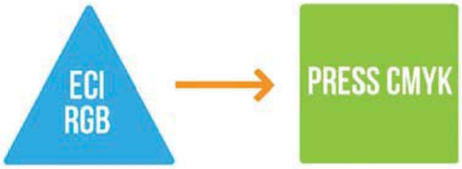


Fig. 4: Color transformations without transparency are relatively simple.

Now consider the same example where the two images are within a "transparency group" in the PDF file. In most cases that group will have a color space associated with it called the "blending color space", and in most cases that blending space will be sRGB, if only because that's the default in many design applications. In addition a "blend mode" will be set. The blend modes allowed in PDF match those shown in Adobe® Photoshop®, including commonly used modes such as 'Normal', 'Overlay' and 'Multiply' and more specialized ones such as 'Soft Light' and 'Saturation'. The colors of each pixel now need to be transformed from the source RGB (ECI RGB) to the blend color space (sRGB).

Once in the blend space the two images need to be composited together. It's unlikely that the pixels of the two images are exactly aligned, so this composition means that the number of apparent pixels in the area where they overlap will increase.

And finally the resulting colors in sRGB must be transformed to the output CMYK of the press.

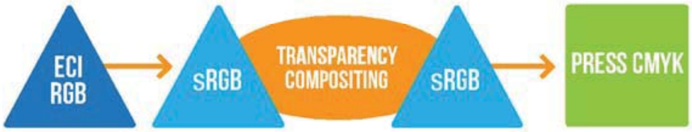


Fig. 5: color transformations with transparency requires significantly more processing.

As you can see this process at least doubles the amount of effort required in color transformations, even without taking into account the work to perform the transparency blending itself, which is significant for some of the blend modes.

The impact of using transparency in a VDP job depends on whether it's used in a 'background' graphic that's used many times on many pages, or if it's in variable data or a re-used object that overlays variable data. If it's in the background the VDP optimizations in many solutions will mean that it only needs to be processed once, which resolves the transparency. The result of that processing can be re-used multiple times so the extra work required in processing doesn't add all that much to the total job time. If it's used in variable data or an object that overlays variable data then the VDP optimizations in many DFEs will be circumvented and the whole of the page may need to be processed as it stands without being able to re-use some or all previously processed elements.

The bottom line on transparency is that it's very valuable, but if it's not in the static background to pages and it can be easily avoided without changing the final printed appearance then do so.

8.3.1 DON'T FLATTEN TRANSPARENCY

It may seem strange after the previous paragraph to say that transparency shouldn't be flattened. But flattening transparency upstream of the DFE can have two significant unwanted effects:

> The transparency effect can sometimes be replaced with a huge number of very small graphics in order to try to maintain exactly the same visual appearance. This not only bloats the file size, but it can make the job even slower to RIP than working from the live transparency would.

> If the flattening is not performed with a detailed knowledge of

the resolution and other capabilities of the press the job will be output on it can introduce some unpleasant artifacts in the output, such as jaggies. Even if you do know the full details for the press that will be used, a pre-flattened job would be harder to transfer to another press at the last minute if you needed to.

8.3.2 AVOID INVISIBLE TRANSPARENCY EFFECTS

Live transparency in PDF is probably most commonly used for drop shadows, but even that use should be avoided if it doesn't result in an effect that's visible on the final printed piece. For example, do not include drop shadows on images that are printed on a black background unless the shadow will also fall on another element where it will be visible, such as another image on the page.

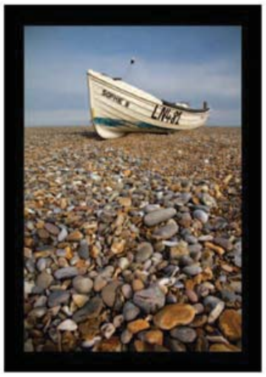


Fig. 6: This image has a drop shadow on it, but it's completely lost against the black background.

Clearly there are exceptions to this where the drop shadow would still be visible on a print, even if it's not on a computer monitor, such as where the drop shadow paints in a rich black (e.g. black plus 40% cyan) and the background is printed with only black ink.

In the same way, if all you're doing is adding drop shadows to text or images that fall entirely on a white background, you don't need to use transparency at all; a simple shading pattern will do everything that you need. Of course, if any of the graphics with drop shadows overlap each other you will need to use transparency, so that the shadows fall across the elements behind correctly.

If assets are being created in off-the-shelf design tools and then integrated with variable elements in a composition tool this may be a difficult optimization to perform because many design tools offer a simple switch to add a drop shadow, which includes turning on the transparency. On the other hand, if everything is created and laid out within the composition tool it should be very achievable.

8.3.3 USE OVERPRINTING INSTEAD OF TRANSPARENCY FOR BLACK TEXT AND RULES

Printers using offset lithography and other conventional print technologies have used a little trick to avoid registration errors between small black text and fine rules running over other graphics on a page for many years: they set the black elements to overprint. This means that the text and rules don't knock out of the other graphics, which means that you'll never see any white outlines as a result of misregistration. More recently we've seen a few instances where people have used transparency instead, using Overlay or Darken blend modes.

The potential for objectionable artifacts when using either approach is disappearingly small. The only visible effect likely is that the black won't be pure, but may have varying amounts of cyan, magenta and yellow behind it. If these techniques are used only for small black text and rules then it's hard to see that variation at all, even with a lens.

Where overprinting and transparency do differ, however, is in the speed at which the DFE can process them. A simple black overprint will often be very significantly faster, especially if the background behind the black elements is complex or includes high-resolution images.

8.3.4 USE CLIPS RATHER THAN MASKS

Clipping an image, either to a smaller rectangle or to a more complex shape, can be done in several ways, and these vary greatly in efficiency:

a) a vector clip-path is by far the most efficient and should be used wherever possible

b) if the creation workflow is such that a vector clip-path cannot be applied, then use a masked image (an image with a Mask entry)

c) by far the most expensive in processing power is a soft mask (SMask), which is the only one of the three approaches that uses live transparency.These should only be used where a soft blend is required, e.g. between an image and a special effect frame.

Some applications use a soft mask to clip an image only because a hard mask at the same resolution as the main image would result in visible stepping around the edge.A vector clipping path will yield a smoother edge than most hard masks and would be a suitable alternative to a soft mask in most cases.

When a special effect frame is added to an image then it is usually printed on top of the image. It is far more efficient to reveal the real image through the frame using one of the following techniques than to add a soft mask to a frame supplied as an image:

a) Draw the frame using vector objects (far easier for some visual effects than for others). In this case nothing extra is required to reveal the image through the center of the frame

b) Apply a clipping path to the frame object

c) Use a masked image (with a Mask entry) rather than an image with a SMask entry.

When using a frame with a complex irregular or non-rectangular shape that requires portions of the real image to be hidden so that they are not visible outside the frame, a clipping path should be used on the main image data as well. This often requires only a relatively rough outline as the clipping path only needs to fall somewhere through the area covered by the frame and does not need to track its edge exactly.

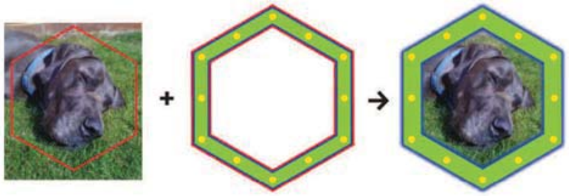


Fig. 7: Clipping images instead of using transparency.

8.3.5 PRE-COMPOSITE IMAGES WITH SOFT MASKS

Some VDP designs include the placement of one image with a soft mask over another background image, perhaps to achieve a soft transition from one to another. If it is possible to composite the two images with the soft mask into a single image before delivery to the DFE, the work required in the DFE will be greatly reduced.

There is little benefit to be gained from compositing multiple images without masks simply because they fall on the same page or because they overlap each other. The coalescing step of the VDP optimization will normally achieve this stage quite efficiently.

8.3.6 AVOID USING TRANSPARENCY FOR IMAGE GHOSTING

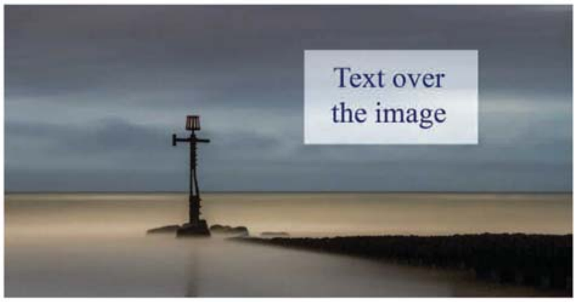
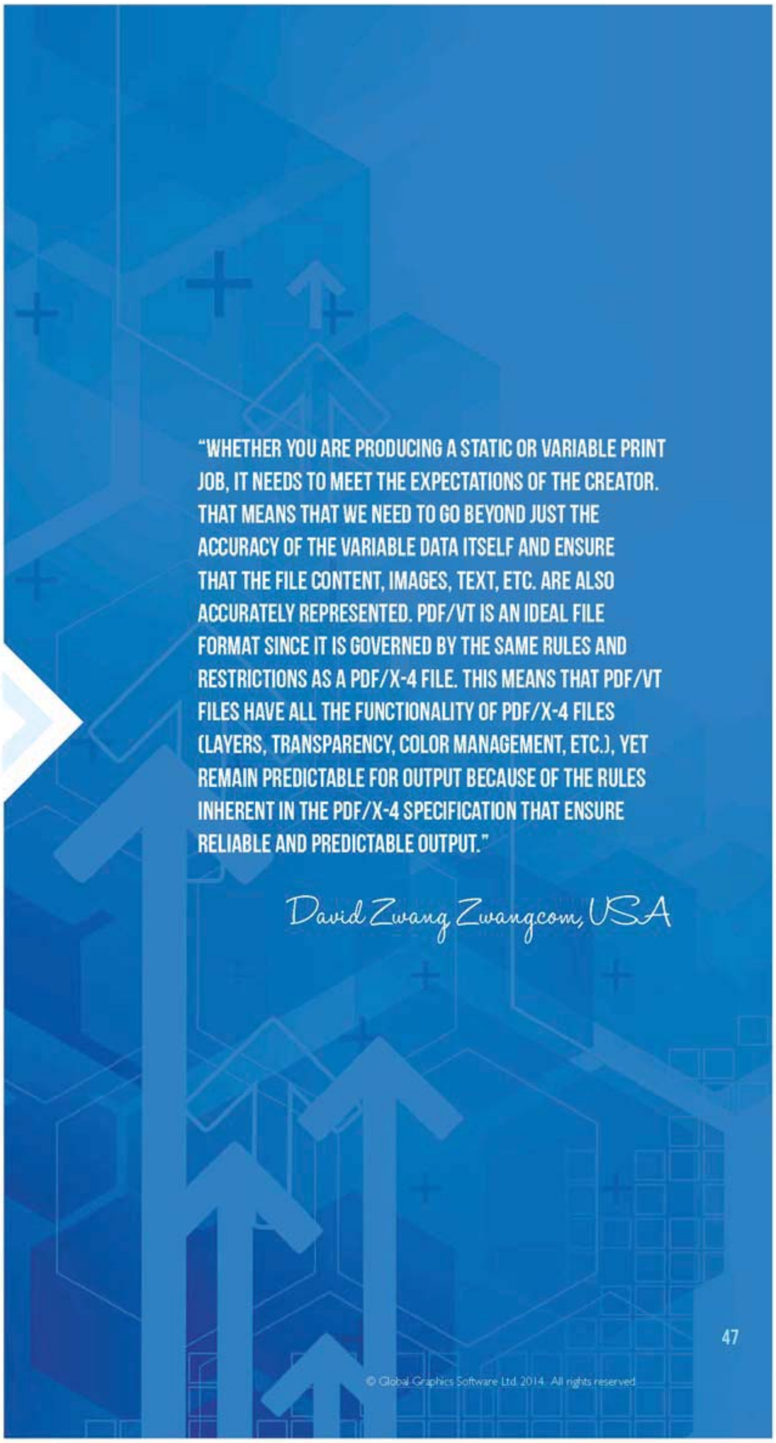


Fig. 8: Ghosting images to allow text on top of them to be read.

One effect that is sometimes used when placing a text block on top of an image is to 'ghost' the image behind the text, reducing its contrast and making it lighter so that the text can be read more easily. This can be achieved by placing a transparent rectangle over the image and behind the text, but that will mean that processing in the DFE will be very inefficient because it needs to resolve the live transparency. Either of these two techniques would more efficient:

a) if every use of the image requires the same size and position of ghosted area then the image and the ghosted area should be pre-composited, resulting in a single image and no transparency in the PDF

b) if the size of the ghosted area must vary for different recipients (e.g. because their address is printed in that space, and addresses differ in the number of lines) then it is better to include two copies of the image data, once for the full background, and once for the ghosted area. The image used for the ghosting may be pre-adjusted before inclusion in the PDF, or the adjustment may be applied using a transfer function.



"WHETHER YOU ARE PRODUCING A STATIC OR VARIABLE PRINT JOB, IT NEEDS TO MEET THE EXPECTATIONS OF THE CREATOR. THAT MEANS THAT WE NEED TO GO BEYOND JUST THE ACCURACY OF THE VARIABLE DATA ITSELF AND ENSURE THAT THE FILE CONTENT, IMAGES, TEXT, ETC. ARE ALSO ACCURATELY REPRESENTED. PDF/VT IS AN IDEAL FILE FORMAT SINCE IT IS GOVERNED BY THE SAME RULES AND RESTRICTIONS AS A PDF/X-4 FILE. THIS MEANS THAT PDF/VT FILES HAVE ALL THE FUNCTIONALITY OF PDF/X-4 FILES (LAYERS, TRANSPARENCY, COLOR MANAGEMENT, ETC.), YET REMAIN PREDICTABLE FOR OUTPUT BECAUSE OF THE RULES INHERENT IN THE PDF/X-4 SPECIFICATION THAT ENSURE RELIABLE AND PREDICTABLE OUTPUT."

David Zwang Zwang.com, USA

For the maximum performance gain, the parts of the image that never fall within the ghosted area may be discarded in the second copy of the image, rather than just clipped out. Care must be taken however to ensure that the two images are exactly aligned in that case. Even though this technique increases the amount of image processing required, it can increase overall performance because image processing is much faster than transparency compositing.

8.3.7 AVOID UNNECESSARY COLOR SPACE CONVERSIONS FOR TRANSPARENCY

As mentioned above, a transparency group in the PDF file can have a blending color space defined within it. In these cases the colors of graphics within the group must be transformed from their original color space into the blending color space, and then subsequently into the output device color space.

Many PDF files have transparency groups with a blending color space set to sRGB, simply because that's the default in a number of mainstream design tools, while the output device color space for print is usually CMYK (or some variant upon that). The transparency doesn't add any additional transformation of the color information if the blend color space of the group matches either the source color space of all graphics within the group or the device color space. The transforms may occur at a slightly different place during processing, but the same amount of transformation is required.

But if neither the source color space nor the device color space match the blending color space the colors of all graphics must be transformed twice instead of once, increasing the overall processing time.

If you can ensure that all graphics (especially images) within a group have the same source color space as the blending space, or, even

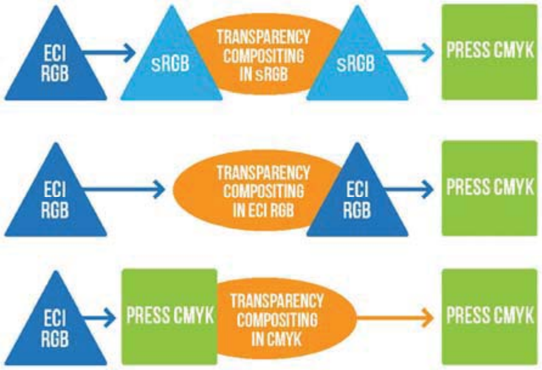

ECI RGB
sRGB
TRANSPARENCY COMPOSITING IN sRGB
sRGB
PRESS CMYK
ECI RGB
TRANSPARENCY COMPOSITING IN ECI RGB
ECI RGB
PRESS CMYK
ECI RGB
PRESS CMYK
TRANSPARENCY COMPOSITING IN CMYK
PRESS CMYK

Fig. 9: Choosing the blend color space carefully can greatly reduce color transformations required.

better, the blending space matches the output device color space, then throughput in the DFE will be higher.

Switching the blending color space, especially between RGB and CMYK spaces, will often change the final printed color. If you're going to change the blend color space from something like sRGB to the output CMYK for maximum DFE performance you need to make that decision early in your design process and ensure that the resulting output is approved. If you need to stay with blending in RGB you should ensure that the blend color space matches the source color space of all of your images (or vice versa).

Occasionally transparency group operations may be chained together if a group is defined within another group, although that is relatively rare. There can be good reasons for using this kind of construct in commercial print, publication or newsprint work, such as when placing or imposing multiple PDF/X files created for the same characterized print condition, but using different ICC profiles in their output intents together. This might arise if you're placing display ads, for instance. If that kind of situation occurs in a VDP print job, however, you would be advised to review the creation workflow and unify your asset design process further upstream to ensure consistent and predictable output. In general nesting transparency groups should be avoided for VDP.

8.4 OPTIMIZING VECTOR GRAPHICS

Vector graphics are relatively quick to process compared to images, which is why this section is so short.

8.4.1 BARCODES AND QR CODES

QR Codes and other barcodes can be represented on a PDF page in several different ways, including as an image or using vector graphics. One dimensional barcodes can also be drawn with a barcode font.

In terms of processing speed a barcode font is typically the most efficient, but can limit the opportunities for compensating for edge growth to maximize readability. Using an image (or imagemask) is generally slowest, so the best compromise tends to be to use vector graphics.

Composition vendors can assist here by turning on automatic stroke adjustment for bar codes (using the SA graphics state parameter in the PDF) to minimize issues if the scaling is not absolutely correct.

8.4.2 AVOID UNNECESSARY SMOOTH SHADES

Smooth shades were added into the PDF specification in the late nineties, and provide a way of defining a variety of graduated tints or vignettes. They can be very useful, but tend to take a little longer than a simple flat fill to process, especially if they happen to interact with any transparent graphics on the page.

Don't use a smooth shade where the final color doesn't vary across the object; just use a flat tint instead.

8.5 OPTIMIZING VDP LAYOUTS

As mentioned above the ability to coalesce multiple graphics together to reduce the number of components that need to be re-composed together to form a final page can have a very significant impact on the throughput of the DFE. The coalescing process typically requires that multiple graphics must all appear on a significant number of pages together, and with exactly the same positions relative to each other in order to be grouped together into a single component.

Some systems have the capability to adjust the drawing order of the assets and other graphics placed on the page, that is the order in which they are to be placed, with some behind or in front of others. Being able to re-order graphics allows them to be coalesced into groups even if they are not adjacent to each other in the drawing order. Of course, those solutions place great importance on avoiding any changes to the visual appearance of the printed page as a result.

Most of the recommendations in this section are aimed at maximizing the efficiency of the coalescing process so that fewer components are required to construct every final page.

8.5.1 PLACE GRAPHICS IN CONSISTENT LOCATIONS WHENEVER POSSIBLE

If you're creating several related page layouts that use the same assets (e.g. images) you may be able to generate each one by copying the previous one and making the necessary changes, or you may need to build each from scratch. In either case you can improve the efficiency with which the final job passes through the DFE by ensuring that there are no unintended changes to the position of each asset on the page as you do so.

If you have a good reason to move things around on the page then go ahead, but finding that the throughput of the DFE is reduced because you accidentally didn't place them in exactly the same position would be frustrating!

In the same way, some composition engines offer the capability to 'flex' layouts, to move some assets in response to differing sizes of something like a text block because some recipients have longer names or addresses, or the length of a list of items varies. Again, if that produces the exact visual result that you're looking for go ahead and use the option. If flexing the layout doesn't provide a benefit for you in the design or readability, turn it off and allow the job to process a bit faster at the print stage.

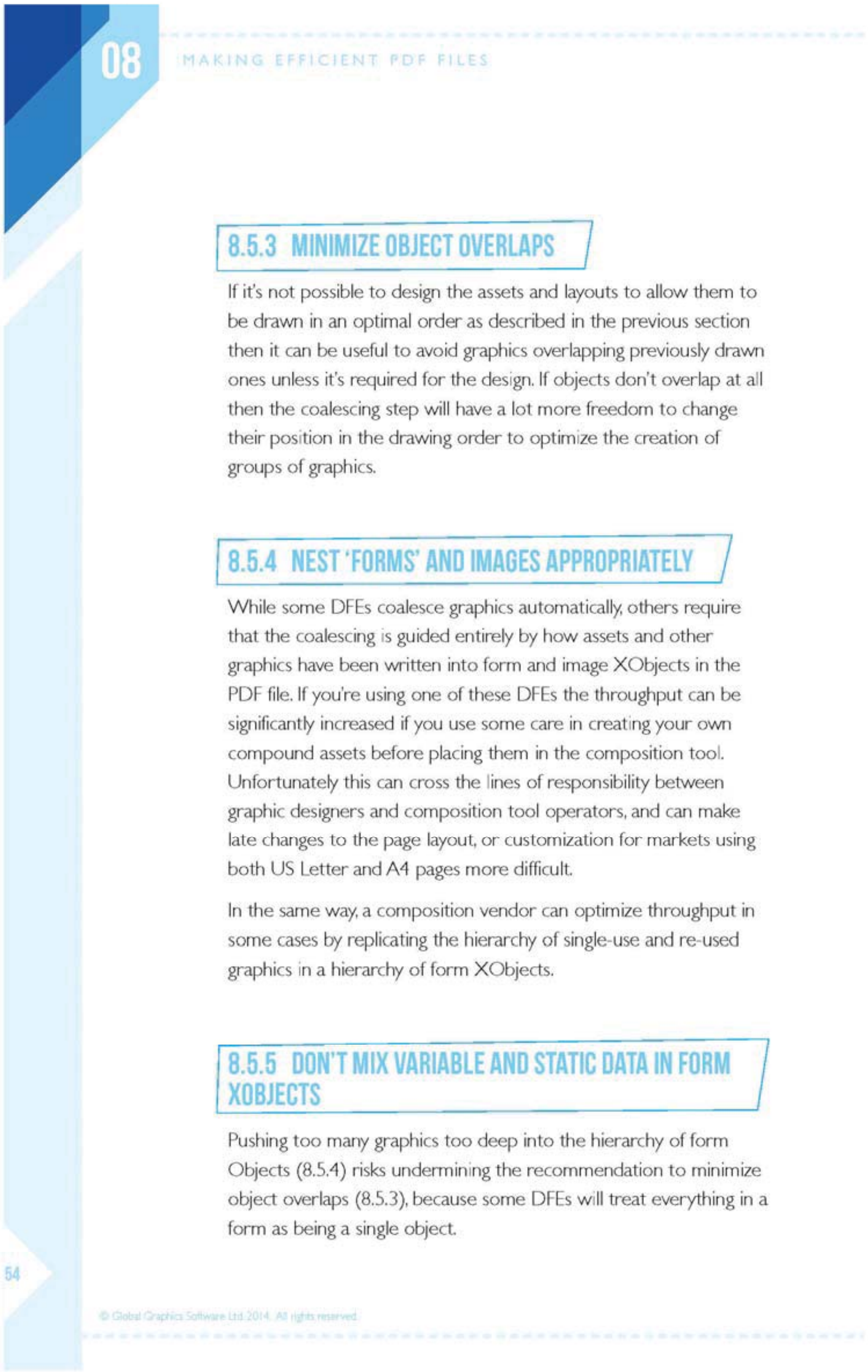
8.5.2 AVOID INTERLEAVING STATIC AND VARIABLE ELEMENTS ON A PAGE

Many VDP designs boil down to a static 'background' that is used exactly the same on many pages, with variable data laid over the top of it, varying by the recipient of that instance. The variable data may be specific to that recipient (e.g. their name and address). Some may also be "semi-variable", where metadata about the recipient is used to select from a relatively small set of options (e.g. a logo for membership level, a map to their nearest store location, etc).

The coalescing process will typically work best if it can merge all of the assets and other graphics for the 'background' into one or a small number of components to be re-composited later. It may collect sets of semi-variable assets and elements together as well, if they are used together in a consistent way. To take the example given above, of a map to the recipient's nearest store, it may be that that map is always used with a logo and a text address for that specific store, and with a sales representative's image and telephone number.

It's common to see PDF files where the assets and graphics on a single page are drawn onto the page in a fairly arbitrary order, so that 'background' graphics are actually drawn quite late, after many of the variable and semi-variable graphics. This often makes no difference to the visual appearance as long as the graphics drawn later don't fall on top of those drawn earlier. But it does mean that the coalescing step must work harder and may not be able to collect graphics into a small number of large components, typically reducing throughput.

If you can design your assets and layouts in such a way that static background elements are drawn first, followed by semi-variable graphics, and then those specific to the current recipient then the coalescing stage can often perform better. At a slightly more detailed level, it's often worth trying to make sure that an image and the key line for that image are next to each other in the drawing order.



8.5.3 MINIMIZE OBJECT OVERLAPS

If it's not possible to design the assets and layouts to allow them to be drawn in an optimal order as described in the previous section then it can be useful to avoid graphics overlapping previously drawn ones unless it's required for the design. If objects don't overlap at all then the coalescing step will have a lot more freedom to change their position in the drawing order to optimize the creation of groups of graphics.

8.5.4 NEST 'FORMS' AND IMAGES APPROPRIATELY

While some DFEs coalesce graphics automatically, others require that the coalescing is guided entirely by how assets and other graphics have been written into form and image XObjects in the PDF file. If you're using one of these DFEs the throughput can be significantly increased if you use some care in creating your own compound assets before placing them in the composition tool. Unfortunately this can cross the lines of responsibility between graphic designers and composition tool operators, and can make late changes to the page layout, or customization for markets using both US Letter and A4 pages more difficult.

In the same way, a composition vendor can optimize throughput in some cases by replicating the hierarchy of single-use and re-used graphics in a hierarchy of form XObjects.

8.5.5 DON'T MIX VARIABLE AND STATIC DATA IN FORM XOBJECTS

Pushing too many graphics too deep into the hierarchy of form Objects (8.5.4) risks undermining the recommendation to minimize object overlaps (8.5.3), because some DFEs will treat everything in a form as being a single object.

For the graphic designer or composition operator this means that graphics that are only used for a single recipient should not be bundled into the same asset as graphics that are used many times for multiple recipients.

8.5.6 DON'T DRAW THE SAME GRAPHIC MULTIPLE TIMES

It may seem obvious that drawing the same graphics in exactly the same place on the same page multiple times may impact on performance, either directly or by reducing coalescing efficiency.

But it's something that we see quite often.

The same comment goes for drawing graphics and then hiding them completely with another graphic over the top. We've even seen cases where a complete page was drawn and then (we assume) the designer or composition operator decided to redo it, placed a white rectangle over what they'd done already to hide it and drew another complete page to replace it. The RIP will still need to do a reasonable amount of work to process the hidden first page, and it's just going to slow things down.

We recommend that you don't be that guy!

8.6 OPTIMIZATIONS IN VDP WORKFLOW SOFTWARE

The recommendations above are relevant for graphic designers and composition operators in at least some workflows. But there are some optimizations that can only be addressed by the software vendors involved, either in asset creation and management, or in the composition tools themselves. These tend to be deeper into the technical aspects of exactly how a PDF file is constructed.

8.6.1 EMBED EACH IMAGE IN THE PDF JUST ONCE

The data for images is embedded in the PDF file as an XObject. The description of the graphical contents of each page then includes a pointer to the XObject to place that image on that page. If the same image is used many times within a single PDF file then the image data can be embedded many times, or it can be embedded just once and the pointer from the page descriptions can all point to that same copy.

If multiple copies of the same image are embedded in the PDF that will evidently bloat the file size. Less obviously it will reduce the efficiency of the VDP optimizations in some DFEs because the images will be seen as different and therefore each copy may be processed separately, increasing the work required unnecessarily and slowing the job down.

Whenever possible only one copy of each image should be embedded. If the same source image is used at multiple different sizes on the pages those may either use the same embedded copy or a separate copy at a suitable resolution may be used for each final size.

8.6.2 DON'T TILE OR STRIPE IMAGES

A couple of decades ago it was common to write images into page description languages as a series of rectangular tiles, or as strips. DFEs and RIPs at that time didn't have access to much RAM, and the intention was to ensure that the RIP didn't need to hold very large amounts of image data at the same time. RAM costs are still a factor in DFE design but the amounts now used are many times higher than they were back then, so this 'workround' is no longer required.

On the other hand, there is a measurable cost for the RIP to set up and tear down a processing pipeline for each image, so making the DFE handle a large number of small images instead of a single large one makes it run slower.

One extreme example of inefficient practice can often be seen when an image has been placed on a page in a design application and then a single color in the image has been marked as transparent by the user. Some applications will generate a huge number of very small images, often in strips only one pixel tall, in the page description language. If they were to including the whole image as one, and using a stencil mask or color key mask on that image it would increase processing speed in the DFE hugely.

And that slow-down is sometimes multiplied by encoding the image strips as in-line images instead of image XObjects. In-line images make it harder for the RIP to separate processing images from that of the rest of the graphics within a page and therefore subvert some of the optimizations that might otherwise be applied.

8.6.3 USE A CONSTANT OPACITY RATHER THAN A SOFT MASK WITH CONSTANT VALUES

There are two ways of specifying how transparent a graphic should be within a PDF file: you can set a constant opacity value for fills and strokes (using the CA/ca keys), or you can attach a soft mask (SMask in the PDF, or within a JPEG2000 image). Soft masks can be very useful if the transparency should vary across the graphic, e.g. for softening the edges of an image. But we've also seen them used quite a lot where the transparency is uniform across the whole graphic. The most inefficient examples add a soft mask where all of the values are either 1.0 (indicating that the element is fully opaque) or 0.0 (indicating that the element is fully transparent, and should not be visible at all).

If the element should be fully opaque the best way to represent that is to omit the SMask entry completely, or to set it to /None.

If the element should be fully transparent (not visible) then don't include it in the PDF file at all!

And if the element should have a constant transparency that is neither fully opaque, nor fully transparent, just use the CA or ca keys to set that value and omit the SMask key or set its value to /None.

8.6.4 DON'T SUBSET FONTS

Some software subsets fonts when embedding them within a PDF file. It's a technique that was originally developed to reduce file sizes slightly and to make it marginally harder to copy fonts by extracting them from PDF files. The incremental increase in file sizes to include a whole font in a VDP file is now trivial compared to disk sizes and communications speeds, with the possible exception of multi-byte fonts, for Japanese or Chinese for example. And most font vendors have adopted different models for font sales that don't rely on avoiding embedding them completely. So most of the advantages of subsetting fonts have disappeared.

On the other hand there are distinct costs from subsetting fonts in a VDP job if that is performed per page. Each subset of the font will be regarded by many RIPs as a different font. That means that the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly.

So we recommend that you don't subset fonts in a VDP job or, if you do subset, you embed a single subset that includes all of the glyphs used on all pages for all recipients.

If, however, you're generating personalized instances of a PDF file for web or mobile device delivery you may want to continue subsetting embedding fonts for each instance, especially if using multi-byte fonts.

8.6.5 USE PDF/VT HINTS WHEN POSSIBLE

If you're making a PDF/VT file then each graphic in the file (expressed as a form or image XObject) can have one or more 'hints' associated with it. This allows the file to carry information about whether that graphic is used only once or multiple times. It also allows it to say that the asset will look exactly the same every time it's used, or if it may be affected by other graphics around it. These hints can provide a short-cut for the DFE's optimizations, allowing it to make decisions more quickly.

If the information is available to set these hints correctly then you are recommended to do so, but do not set the hints if you're not confident that you will get them right.

ABOUT MARTIN BAILEY

MARTIN FIRST JOINED WHAT HAS NOW BECOME GLOBAL GRAPHICS SOFTWARE OVER TWENTY YEARS AGO, AND HAS WORKED IN CUSTOMER SUPPORT, DEVELOPMENT AND PRODUCT MANAGEMENT FOR THE HARLEQUIN RIP AS WELL AS BECOMING THE COMPANY'S CHIEF TECHNOLOGY OFFICER. DURING THAT TIME HE'S ALSO BEEN ACTIVELY INVOLVED IN A NUMBER OF PRINT-RELATED STANDARDS ACTIVITIES, INCLUDING CHAIRING CIP4, CGATS AND THE ISO PDF/X COMMITTEE. HE'S CURRENTLY THE PRIMARY UK EXPERT TO THE ISO COMMITTEES MAINTAINING AND DEVELOPING PDF AND PDF/VT.

THANK YOU TO OUR SPONSORS...

DEVELOPERS OF SOFTWARE PLATFORMS FOR DIGITAL PRINTING, DIGITAL DOCUMENT AND PDF APPLICATIONS

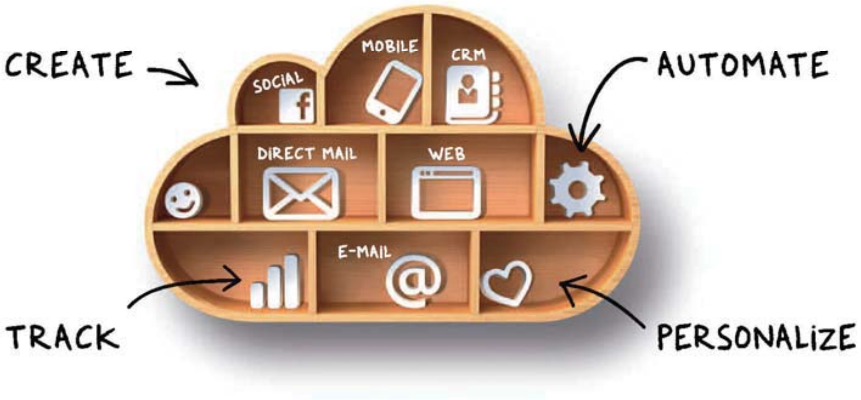
GLOBALGRAPHICS.COM

GLOBAL GRAPHICS® software

DirectSmile®
CROSSMEDIA MADE EASY

All-in-one Software for

Marketing Automation



DirectSmile allows you to create, personalize and automate marketing across web, e-mail, mobile, social and print media. It doesn't require any programming skills, and it integrates with your existing CRM system.

For corporates

Automate your marketing processes and win more leads through your website. Unleash the potential of your marketing team.

For agencies

Secure recurring revenue with automation. Design, personalize and manage marketing campaigns across all media.

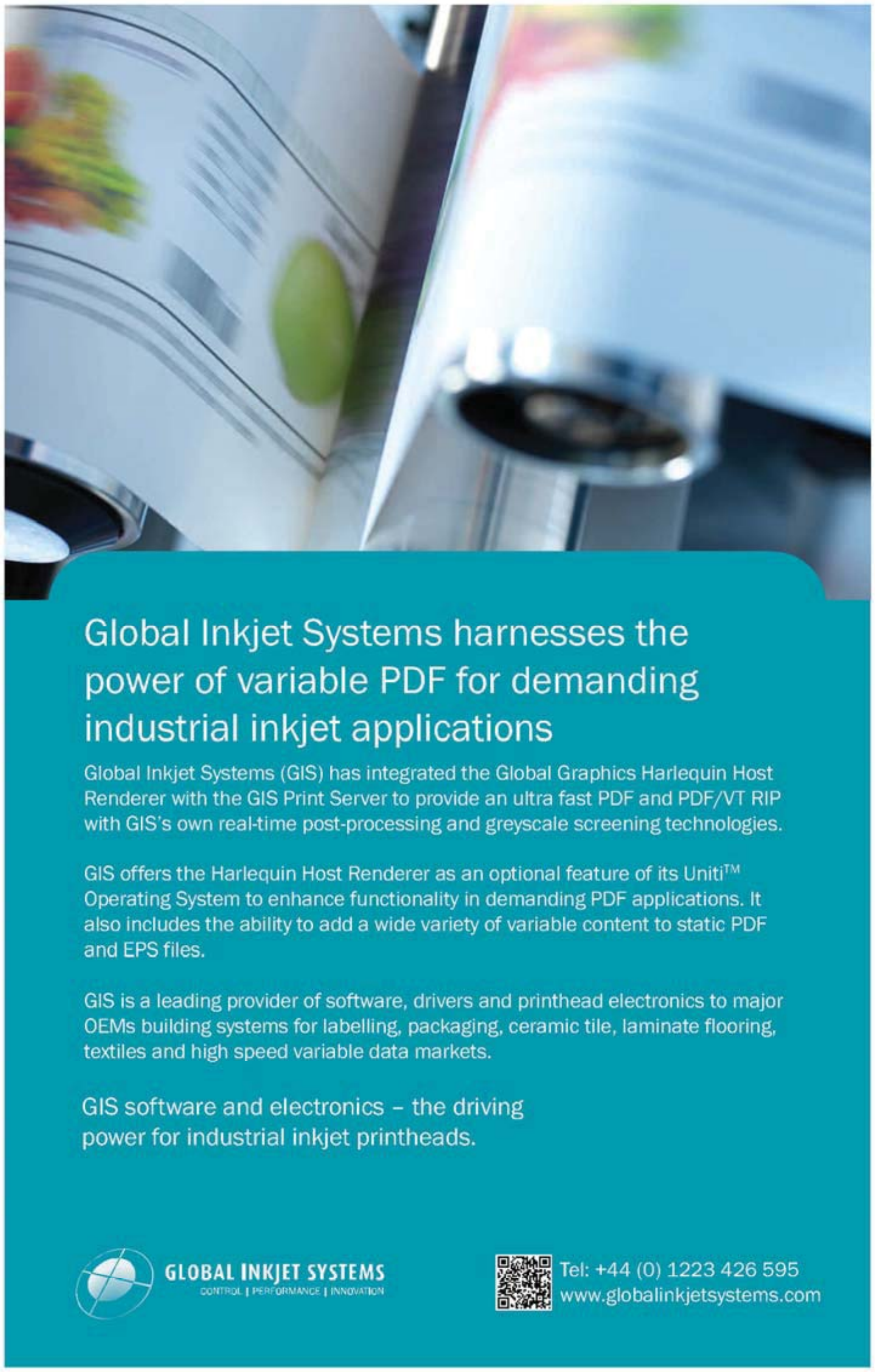
For printers

Build your own web-to-print and print-on-demand applications and produce variable data printing products.

For designers

Create data-driven web applications without any backend programming. On the frontend, you can work with HTML or without.

Phone (World): +49 (0) 30 62 777 111 | Phone (US): 973 338 9200 | sales@directsmile.com



Global Inkjet Systems harnesses the power of variable PDF for demanding industrial inkjet applications

Global Inkjet Systems (GIS) has integrated the Global Graphics Harlequin Host Renderer with the GIS Print Server to provide an ultra fast PDF and PDF/VT RIP with GIS's own real-time post-processing and greyscale screening technologies.

GIS offers the Harlequin Host Renderer as an optional feature of its Uniti™ Operating System to enhance functionality in demanding PDF applications. It also includes the ability to add a wide variety of variable content to static PDF and EPS files.

GIS is a leading provider of software, drivers and printhead electronics to major OEMs building systems for labelling, packaging, ceramic tile, laminate flooring, textiles and high speed variable data markets.

GIS software and electronics – the driving power for industrial inkjet printheads.

GLOBAL INKJET SYSTEMS
CONTROL | PERFORMANCE | INNOVATION

Tel: +44 (0) 1223 426 595
www.globalinkjetsystems.com

Start a new chapter in publishing

Publishing is changing. HP Digital Printing lets you tackle the changes with real confidence. With our sheet- and web-fed digital presses, you can handle high-value and high-volume jobs. So from manuals and textbooks to glossy magazines and newspapers, you'll deliver print to be proud of – chapter after chapter, cover to cover.

Find out more at hp.com/go/publishing or call 800-289-5986.

Game changers

Say yes to 98% of commercial jobs, produced cost-effectively with no compromise on quality or media range.

Introducing the latest ground-breaking innovations from HP Indigo. Faster, smarter and more profitable, this new line-up is changing the rules of the game. It includes the HP Indigo 5600 and 7600 digital printers, as well as the B2/29 inch HP Indigo 10000 – all you would expect from an Indigo press, but in a wider format, it's a game-changing portfolio that's redefining what's possible with print.

With thousands of digital sheetfed patterns in operation, as well as partnerships with leading vendors for end-to-end solutions, HP Indigo is the proven choice for profitable growth.

Contact your Account Manager
or visit us online at hp.com/go/indigo

HP Indigo 7600 Digital Press

HP Indigo 5600 Digital Press

HP Indigo 10000 Digital Press

PODi

The leading independent global organization supporting digital print since 1996

Our goals are to:

- Grow the digital print market by educating marketing professionals and print service providers
- Build a community of professionals who see the value in openly sharing ideas
- Help members build successful businesses using digital print

PODi

the Digital Printing Initiative

One of PODi's primary objectives is to enable efficient variable data printing for marketing applications through support for standards like PDF/VT and PPML

"As a leading PODi Member and supporter of our goals, Global Graphics has produced an invaluable guide to optimizing PDFs for personalized digital print and automated on-demand workflows. Its many practical recommendations and technical explanations will be immensely useful for PODi Members and the whole industry of service providers, content creators and software solutions vendors engaged in digital printing".

Tony Hodgson
Director for PODI Europe

www.podi.org

SCREEN

A new standard in high-speed inkjet printing

Truepress Jet520

☑ 70 years of colour printing experience

☑ Up to 3,000 pages per minute

☑ Highest quality digital print

☑ Industry leading uptime

On-demand. For all your demands.

Take a closer look at all of our print-on-demand solutions. With the experience of more than seven decades in developing printing equipment, SCREEN is recognised as a world leader in image-processing technology—we make high-class printing a reality. In delivering advanced solutions that answer user needs like no one else, our know-how and expertise truly cover the printing world from A to Z. Our print-on-demand machines not only support current printing solutions, but ensure a strong future presence as new business opportunities arise.

Contact us at sales@screeneurope.com

GLOBAL GRAPHICS® software

www.globalgraphics.com

Distributed by:

WITH THANKS TO OUR SPONSORS

DirectSmile® CROSSMEDIA MADE EASY

GLOBAL INKJET SYSTEMS

hp Indigo

PODi the Digital Printing Initiative

SCREEN

Xplor® International

hp Inkjet High-speed Production Solutions

Document Designated Attorneys’ Eyes Only: Redacted Pursuant to Protective Order

Appendix Pages A1575-1604

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

In Re:	)	**Case No. 3:15-md-2614-M**
	)	
INDUSTRIAL PRINT TECHNOLOGIES, LLC,	)	Dallas, Texas March 29, 2016 1:30 p.m.
Plaintiff,	)	
v.	)	STATUS CONFERENCE RE: - EMERGENCY MOTION TO COMPEL DEFENDANTS TO IDENTIFY EMAIL CUSTODIANS [36] - SEALED MOTION TO COMPEL DEFENDANTS TO PROVIDE FACT DISCOVERY [50]
CENVEO, INC, HEWLETT-PACKARD COMPANY, ET AL.,	)	
Defendants.	)	

TRANSCRIPT OF PROCEEDINGS
BEFORE THE HONORABLE PAUL D. STICKNEY,
UNITED STATES MAGISTRATE JUDGE.

APPEARANCES:

For the Plaintiff:

Alison Aubry Richards
FITCH EVEN TABIN & FLANNERY, LLP
120 South LaSalle Street,
Suite 1600
Chicago, IL 60603
(312) 577-7000

For the Plaintiff:

Steven C. Schroer
FITCH EVEN TABIN & FLANNERY, LLP
1942 Broadway, Suite 213
Boulder, CO 80302
(303) 402-6971

For the Defendants:

Audrey Maness
WEIL, GOTSHAL & MANGES, LLP
700 Louisiana, Suite 1700
Houston, TX 77002
(713) 546-5000 x5317

For the Defendants:

Edward R. Reines
WEIL GOTSHAL & MANGES, LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065
(650) 802-3022

For the Defendants:

M. Brett Johnson
FISH & RICHARDSON
1717 Main Street, Suite 500
Dallas, TX 75201
(214) 292-4007

Court Recorder:

Lavenia Price
UNITED STATES DISTRICT COURT
1100 Commerce Street, Room 1452
Dallas, TX 75242-1003
(214) 753-2168

Transcribed by:

Kathy Rehling
311 Paradise Cove
Shady Shores, TX 76208
(972) 786-3063

Proceedings recorded by digital sound recording; transcript produced by transcription service.

DALLAS, TEXAS - OCTOBER 20, 2015 - 1:34 P.M.

THE COURT: Good afternoon. We're here in the matter of Industrial Print Technologies versus Cenveo, Inc., Hewlett-Packard, et cetera, 3:15-md-2614-M. Who's here for the Plaintiff?

MS. RICHARDS: Your Honor, Alison Richards and my partner Steve Schroer for Plaintiff IPT.

THE COURT: Thank you. For the Defense?

MR. REINES: Edward Reines and Audrey Maness from Weil Gotshal and Brett Johnson from Fish.

THE COURT: Thank you.

All right. Now, Judge Lynn has referred this matter over to me, the emergency motion to compel the Defendants to identify email custodians as well as a sealed motion to compel Defendants to provide fact discovery relevant to the variable data patents. She has asked me to keep an eye on this and hold periodic conferences to address ongoing discovery management concerns as appropriate. Hopefully, that means we just meet today and I won't see you again. But my guess is, from the history of this case, that you'll need some help in getting some of these matters addressed and resolved.

My understanding also is that you've had some phone conferences on discovery disputes with Judge Lynn previously and that you were able to work things out, kind of. No?

MR. REINES: I think some progress was made, but really, it didn't break the logjam. I mean, I think there's one basic logjam. I don't know if Your Honor will be able to resolve it all at one. But from the perception of Defendants, there's one logjam, and that was sort of percolating along but really hasn't been tackled at this point.

THE COURT: All right. Ms. Richards, I'm happy to hear from you.

MS. RICHARDS: I don't agree that there's one major logjam.

THE COURT: Okay.

MS. RICHARDS: In particular, we had a conference with Judge Lynn about the ESI order.

THE COURT: Yes.

MS. RICHARDS: And it's Plaintiff's position, as I'm sure the Court is aware from the briefing, that the Court has already ruled on that issue twice. But, nonetheless, we're back here, Your Honor.

THE COURT: All right. Well, tell me what you need today.

MS. RICHARDS: Thank you, Your Honor. Would you prefer that I speak from the table --

THE COURT: No, you're fine there.

MS. RICHARDS: -- or the podium? I guess the easier issue to deal with, from our perspective, is the one I just alluded to relating to the emergency motion relating to the email.

THE COURT: Yes.

MS. RICHARDS: From our perspective, this is a black and white issue, and it's a case of "If Mom says no, ask Dad." The history of the ESI order is that it was negotiated and entered two or three times in one of the member cases in the Eastern District of Texas, re-negotiated here and entered as an order subject to Rule 60 last summer.

At the Rule 16 conference, Defendants asked Judge Lynn to change the order. She declined to do so. Then Defendants got on the phone with the Court on February 23rd, one of the conferences, and asked her to change the order. She declined to do so. What she said was, "I don't see how it would hurt for the parties to meet and confer, and I'm not opposed to IPT identifying some topics, but I'm certainly not shifting the burden to IPT to prove a need for the discovery. Meet-and-conferring never hurts." But then after that she ordered Defendants to identify their custodians. They still didn't do so.

So the situation that we're in is the case management schedule is linked off of the claim construction order. That discovery closes within four months of the claim construction order, which was issued on February 12th. That means fact discovery closes on June 12th. From our perspective, what makes sense, so that we don't have to take depositions multiple times, is to obtain documents, including email, before the individuals' depositions.

For example, a couple weeks ago I took the deposition of the first printer defendant from the Defendant O'Neil, Mr. Al Thorpe (phonetic). He was one of the two email custodians that O'Neil identified. But at the deposition, he identified all sorts of documents related to both infringement and damages issues that we don't have. So it's possible that in another case, in another circumstance, we wouldn't be so adamant about getting email, but we've been trying to get these documents, technical specifications, diagrams. O'Neil communicates with its customers about the file that's sent to O'Neil. O'Neil communicates internally about how they process these documents. The witness told me that O'Neil communicates internally about how the jobs are priced, how often they're done. These are all the documents we've been seeking for years, and that deponent, like HP's deponent, suggested that -- well, more than suggested, testified -- that they're in email.

So that's why we've, you know, we've asked for email since last April. It's now been about a year, we've got four months left, and Defendants are still refusing to identify their custodians. They've identified a few. Cenveo identified one; Cimpress, three; Fort Dearborn, zero, but I believe we've worked that out for the time being; and O'Neil, two. O'Neil is the only deposition that we've yet taken. We're trying to avoid taking -- you know, we've got six defendants, and if you have to take every technical deposition twice and every damages deposition twice, that's quite expensive and burdensome and takes quite a bit of time. So we're trying to get the documents before the depositions. But we went ahead and took O'Neil, and that deponent identified plant managers, corporate officers, sales people, IT people, business operations people, programmers, pricing people, all of whom have these documents that we've been seeking in email, but none of those people were identified as a custodian.

So Defendants have already been ordered several times to provide their custodians, but they still haven't, and we have four months left.

THE COURT: Okay.

MS. RICHARDS: So, that's our basic position on the email motion.

THE COURT: All right. For the Defense?

MR. REINES: Thank you, Your Honor. The way the email discovery was structured in this case is somewhat unusual, at least in my experience, which is, under the Eastern District of Texas model that we sort of inherited here, before the propounding party seeks to specify the custodians they want and the terms they want, the party that's surrendering the discovery has to identify a potential cast of custodians.

THE COURT: Uh-huh.

MR. REINES: And the number here was 15 per Defendant, which would make for about 75 names across all the Defendants. What happened was email discovery was not pursued. That conference was back in June. Email discovery was not pursued until it was mentioned by Plaintiffs in September. We engaged and we said, all right, we're ready to do some exchanges in early October, and then they dropped the issue until January. That's all undisputed.

By that time, they had started to take 30(b)(6) depositions and they had source code on all this information.

THE COURT: Okay.

MR. REINES: The argument that we made to the Judge in February, we approached the Court and said, given the development of the case, issues should be crystalizing, and so just for us to shoot in the dark and name 15 custodians isn't really sensible, it should be on issues. She agreed with that. My accounting is a little bit different than what you just heard. She agreed with that and she directed Plaintiff to specify what topics they thought were appropriate for discovery, and then we could match custodians to the issues, rather than shooting in the dark on the custodians. Okay?

They provided a list of topics which we thought was incredibly broad, but we wanted to get into a dialogue, because that's how these things normally work, and we gave a responsive one that said, look, your topics are way too broad. Here's some -- here's what we think you want and here are some custodians that would actually be knowledgeable. The Plaintiffs --

THE COURT: So basically you're telling her what discovery she wants?

MR. REINES: Uh, --

THE COURT: She can run her own discovery. Why can't you give the list of these custodians to her --

MR. REINES: So, --

THE COURT: -- based on what she requested?

MR. REINES: So what happened was they filed the emergency motion the following day, arguing essentially what Your Honor just said to the Court. And the Court held the teleconference and she said, I don't think you two have met. I think actually email discovery isn't just what they want, that there should be a negotiation and a synthesis of what the crystalized issues were, and she said, so I'm not going to decide it now. I'm also very busy. I want you to meet again, I think the motions are premature, and see if you can come up with what the crystalized issues are, and then identify custodians.

Your Honor, if you could just indulge me for five or ten minutes, I think I can give you background on the case that will just hopefully help you help us.

THE COURT: Okay.

MR. REINES: And I mean that most earnestly. So, there's really two disconnects that we're having, okay? The topics that we got are all about variable data printing, okay, and that's just form letters and all kinds of printing that goes on in time immemorial and is happening in printing places all the time, okay? The patent relates to the actual part when you take the static information -- that is, the form and the names -- and you combine them really late in the process, after the data that you have has been made into bitmaps, which is really the end in something called a RIP. I'm not going to get too technical with you, but that -- so it's really -- so there's all kinds of variable data printing that happens at these companies that has nothing to do and I don't think anyone could say -- it's not like a -- it's not a refined infringement issue. It's just if the parties are merging the data in PDFs and making PDFs, and let's say there's 100 letters and it's 100 PDFs, so each document becomes its own document, and sends it to a press, no one is going to say that's infringing unless there's bitmap reuse that's done at the back, where the static data and the variable data is being collated deep into the press.

So the whole problem we have is that we have companies that are doing all of their variable data printing outside the press. In other words, their job is to make the copies of the different PDFs and then have them introduced to the press after this magic is done where you put the two together.

So what really makes absolutely no sense, and the only -- look, I want to make discovery easy -- is we need to have topics that relate to what the issues are, which is bitmap reuse in the press, or theories, at least, tenable theories of what could be bitmap reuse, and not just, do you use variable data printing, all your custodians related to variable data printing, your sales and marketing teams related to variable data printing. It would be colossal waste to do that. We're trying to get crystallization. Okay?

Now let me just explain the next step. So, when you are giving the document, the electronic document to the press, there are different formats you can use. As in the real world of non-printing, PDFs are the most common. There are other types, but PDF is predominant, to say the least. When standard PDFs are used, there is no bitmap reuse. When optimized PDF is used, that's when it's -- it's sort of a bell or whistle that you can use, and at that point you're in the position that you could be doing bitmap reuse. Okay? So that's just some points.

THE COURT: Okay.

MR. REINES: The next set of points to understand this thing is there's three players in the technology. There's a company called Global Graphics, which is a third party that makes the RIP, and it would be in that RIP engine, in that software, where the actual bitmap reuse takes place. So that's one party.

The second party is HP, and HP manages that RIP with its software. So it can tell that RIP, do bitmap reuse or not do bitmap reuse. That's the second thing.

And then you have the printers, who don't know -- in general, have no idea what's going on, or very limited idea of what's going on in any of that. And it's -- I've been there. It's blue collar people earnestly working hard that are checking a box that says either optimized PDF or not. And to say that they have knowledge about how the inner workings of the RIP is is just farfetched. And so fearing that they have documents about that doesn't make a lot of sense.

Okay. So we have -- let me take one example of O'Neil. So O'Neil gave -- there's interrogatory responses that are 15 pages. I mean, we're not -- if you look at the amount of discovery we've given, 15 pages of narrative explaining how their process works. And what they've explained is we only use PDF. We don't use J-Lite or some of the more exotic types of formats. We only use PDF. And under oath, a 30(b)(6) deposition witness said, "And we only use standard PDF. We don't use optimized PDF."

So, if it's true that standard PDF couldn't possibly involve bitmap reuse, then there's just no infringement theory.

Now, I understand, I've been -- I'm a plaintiff more than

I'm a defendant. I understand not trusting someone. But the issue is all we're asking for is, are you challenging whether standardized PDFs have bitmap reuse? If you are, we can talk about that and we could do discovery about that as appropriate. If -- I mean, we don't think there's any basis for that. If your allegation is that they are using optimized PDF, even though the witnesses are saying they're not using optimized PDF, I could imagine discovery on that topic. I'm so reluctant to propose discovery in a case which I think involves waste, but I could imagine an email discovery topic that says, we want your custodians who might know whether you're using optimized PDF, and search the term "optimized PDF". If it hits "optimized PDF," then we know that you weren't telling the truth when your 30(b)(6) witness swore and when your interrogatory said that you're not using optimized PDF.

But the topics as currently formulated are variable data printing, and that doesn't make any sense because that's -- you know, that's just the whole company of O'Neil. And so this is the problem. Now, we filed the motion yesterday, as we said we would. We just got their amended contentions that said -- their contentions basically say -- are boilerplate for all the customers and say you all use PDF, J-Lite, and they name six different formats, and/or this. Okay? Under patent rules, law that I know of, for your final contentions, it shouldn't be a laundry list of six and you hope something is there. When you have 30(b)(6) deposition testimony and you have interrogatories and there's no documents which are inconsistent with it, you should say it's PDF. If we are fraudulently withholding other information, then they can amend later, but there just has to be some theory.

Now, what is very concerning to us is that in their final infringement contentions, which we got not last week but the week before, they actually just essentially copied the preliminary ones from before discovery started, and it says in the actual document, it says in the document, "This is all based on public information. We have received no discovery at all." So they didn't even bother to update the final infringement contentions they served two weeks ago to reflect the fact that they've had source code for a year, that they've had depositions and so forth.

All we're asking -- we're not just stonewalling or doing something crazy to just -- this drives up costs for us, too. It's just, before we go, especially email, but in general on a variable data printing blunt instrument discovery hunt, we should have contentions. Are you alleging that people are using other than PDF? If so, do you have a basis? Are you alleging that they use optimized PDF, which is what the arguable infringement is according to their theory as we understand it, or not? And just a little bit of shape to it, rather than: we're entitled to everything by variable data printing.

And then the other piece to it is they've actually requested things like they want us to correlate all our documents to their claim requirements. In other words, if they have a claim requirement of doing something, they want us to tell all our documents which would correlate to that. Of course, we deny infringement, but they want us to basically make their in theory for them. And so we filed the motion, as we said we would, which is prompt in view of when we got the final infringement contentions that say, for multiple reasons, these contentions do not frame a case, they don't -- they're inadequate wholly. If we can resolve this where they're asking focused questions on the issues in the case, we'll work with them on discovery. If it's identify 15 people that know about variable data patents in a printing factory, we're all going to waste a lot of time and energy. I don't know why -- I've tried to explain this so many times.

One last piece is there's two press families. One is called indigo and one is called PWP. I think it's web printing. That has the feature of the RIP that could possibly do bitmap reuse, which is called the Vari -- I think it's --

MS. MANESS: Harlequin VariData.

MR. REINES: Harlequin is the name of the RIP and VariData is the feature within there that would let bitmap -- some form of bitmap reuse. We think it's totally different than their claimed one, but forget -- we're not debating that. Some level of bitmap reuse taking place in there. That's completely shut off for that press family. We have witnesses under oath who have said that. There's no documents which suggest that there's any capability for bitmap reuse in a whole family of presses. And they're still seeking discovery on this topic. And at some point -- I mean, I understand they get to ask their discovery and they get to explore, but if there's a press that doesn't have the feature whatsoever and there's -- unless there's an articulated basis to believe that that can be challenged and then if you can challenge it, challenge it in a focused way, but not: we get all discovery regarding this press. We're mid-discovery, when final contentions were due two weeks ago.

You've been more than patient. I know you had a trial all day, so I appreciate you listening to those thoughts.

THE COURT: Thank you. Ms. Richards?

MS. RICHARDS: Thank you, Your Honor. I believe that one of the fundamental disputes that brings us here today is that Defendants' basic position is that IPT is only entitled to carefully-worded representations from counsel and carefully-chosen 30(b)(6) witnesses that all repeat the same sentences about what the press does not do, but we're still looking for the documents about what the press actually does do and how it processes the variable data print jobs.

He said several times that we seek all documents about the presses or all documents about variable data printing. Absolutely not the case. IPT is a small plaintiff. We seek specific targeted discovery. If you look at the topics that we actually served for email, I think they're quite specific. None of them say everything about variable data printing. They seek specific damages, people -- custodians who have email about specific damages issues, custodians who have specific documents about how variable data documents are printed.

Throughout the briefing and this morning Defendants have suggested that this case is limited to the issue of whether these Defendants used this optimized PDF feature. Not the case. As Counsel mentioned, there are several file types processed by these printers, PPML and J-Lite being two. Several of the printer defendants actually use these file types. That's something that we agree on. To date, Defendants have offered absolutely no non-infringement contentions related to those file types. So, as we understand it, they're agreeing that those file types infringe. We still need the discovery about those to meet our burden. We have asked for that specific discovery and we believe we're entitled to those documents, not just statements from Counsel and statements from prepared witnesses about what it doesn't do.

And to illustrate that point, I'd like to give the Court an example. So, you have O'Neil. That's one of the printer defendants. The way that their situation works is they get a file from O'Neil's ultimate end customer and then that file goes to some prepress software applications -- from customer to prepress software application -- and then to the HPDFE. And inside the HPDFE is this Global Graphics RIP, Global Graphics RIP being provided by a third party. So O'Neil's witness tells me he's never heard of Global Graphics, he has no idea what they do, he's never talked to them, he's never seen a specification describing how the RIP works, he has no idea how the RIP works, he's never seen a document about it, but he knows that the RIP doesn't work according to the claims. And that's what Counsel is asking us to accept and put in our infringement contentions and is saying there are no documents that say that that isn't the case. Well, we're just not ready to accept that if the other statement Defendants' counsel made, that the printer defendants don't know how the RIP works, is more accurate.

So the Global Graphics' deposition is coming up. Defendants are forcing us to go to London to take that deposition. But I do believe that that will shed some light.

But the bottom line point is IPT is not required to prove its case before it gets that discovery. That turns the entire process as we understand it on its head.

He mentioned something about they had patent claims saying where in the process the specific printing methods have to occur. They have to be at the back. To keep the record straight, that's not the case.

The other thing that Defendants' counsel stated that's not true is that standard PDFs can't meet the patent claims. O'Neil's witness also made this representation, but we believe it's not true. If you look at the publicly available PDF standards, a standard PDF certainly can meet the patent claims, depending on how it's used. So we're looking for that specific discovery about how the printer defendants use the PDF and how Global Graphics and the other RIPs process it.

With respect to the representation that Defendants have offered narratives now about how the system works, that is true to some extent at this point in the case. There's still a problem of the case schedule and the timing, which I hope we'll return back to. We still have at least ten to fifteen depositions left to take in the case in the three or four months remaining. The discovery is only now coming in.

The other point, returning back to the topics for the email, these topics as Judge Lynn ordered -- oh, by the way, he also said that when she got on the phone to refer the case to you, she made -- Judge Lynn made some statement that Defendants were correct, the issues weren't sufficiently crystalized or the topics weren't sufficiently specific. That absolutely did not happen. Judge Lynn got on the phone and was very nice and welcoming to us, didn't entertain any argument from anyone, and asked us to meet and confer again, give her a status report on all issues in discovery, and asked us to let her know if we'd be acceptable to being in front of a magistrate. That is all that happened on that call, Your Honor.

Counsel also stated that our topics would create some sort of colossal waste, but the topics as written are not as broad as Defendants' counsel would suggest. And, further, the ESI order specifies that, at most, we're entitled to choose eight custodians and ten search terms per custodian. So this idea that it's going to be this huge -- I can't remember the word that was used in briefing -- blitzkrieg or colossal waste -- just isn't the case. That was the negotiated purpose of the ESI order from the beginning, was that email discovery shouldn't be huge but there should be some within limited bounds.

And with respect to this issue, you know, Defendants first raised an issue with our infringement contentions March 15th, about three weeks after we filed the motion to compel. So all these years of history of, you know, O'Neil's witness telling me he did nothing to search for documents, us having a handful of documents from the printer defendants, me taking the HP Able (phonetic) deposition without any documents, they weren't created by IPT's amended contentions or lack of contentions. If you look at the specific local rule, Miscellaneous Order No. 62 in this district, Local Rule 3-6, it requires us and allows us to only amend our contentions based on what is required -- it uses the word "required" -- based on the claim construction order. So we're not allowed to go back and redo our case and add discovery and make adjustments. The only thing we're allowed to do is change the contentions based on the claim construction order. So we made minimal changes. There's neither an allowance for adding discovery nor is there a requirement that we do so. But this leads back to this idea that Defendants' counsel can just make representations about how it doesn't do and have their witnesses consistent with that, and their suggestion, the implication is that we're required to assume that those are true and put them in our infringement contentions no while all this discovery we haven’t been -- we've been asking for for years but haven't gotten hasn't been produced. And we don't think that that's the proper sequence of things, to say the least.

THE COURT: All right. Mr. Reines?

MR. REINES: Thank you, Your Honor. First of all, the topics, which is A-1 in the appendix to the email motion, 1-A, there's pages and pages of them. There's probably 20, 30, 40 topics. Is the printer defendants' capability to process and print variable data print jobs? That's the business of O'Neil. They're -- that's what they print, is variable data print jobs. So it's completely unfounded.

What happened in the call with the Court is the Court said negotiate more and if you can't reach agreement on the right catego... I didn't say that she looked and said yours are wrong and you're right. She said, I think you need to meet more to get crystalized issues and send -- and provide a status report.

THE COURT: And you did that, right?

MR. REINES: And we did -- well, they said that they were entitled to just draft them, that they weren't -- there was no negotiation whatsoever. We're trying to get -- we would just like some shape to this. And Your Honor, what I propose, I have a proposal for how we'd proceed, which is we file the motion -- by the way, March 15th, we complained about their contentions. We got them March 14th. That's when they were due, the final contentions. We also have an interrogatory response which requires updating. In the current motions pending before Your Honor, they argue that our non-infringement contentions -- that is, our responsive explanation of why their infringement contentions fail -- need to be updated, and they -- it's actually a lead argument of theirs, is that they're moving because we haven't been updating them enough. And yet they're saying that their infringement contentions require no updatings in the whole entire discovery period. And we did ask an interrogatory, so it has the same supplementation. It can't be that we have to update the non-infringement contentions when the infringement contentions can stay old. That makes no sense.

So what I propose is we filed the motion last night on infringement contentions which really should frame the case. If Your Honor thinks we're wrong about the infringement contentions --

THE COURT: Yes, but that's not referred to me.

MR. REINES: Because we just filed it last night.

THE COURT: I know, but it may not ever be referred to me.

MR. REINES: It's the basis for --

THE COURT: I understand that, --

MR. REINES: All right..

THE COURT: -- but unless it's before me, I can't do anything for you.

MR. REINES: Well, what I was going to --

THE COURT: I mean, I can read it.

MR. REINES: What I was going to propose is what I think makes sense, Your Honor, just in a practical sense, is Plaintiffs are highly motivated to get this discovery dispute moving, as are we, and nobody wants to spend more time on this what I view as a logjam at this point in time. If they in some whatever time they find acceptable submit an op, we submit a reply, we're willing to do that within two days, and then I -- you -- the Court -- we can ask or someone can ask for this all to be joined together. Because the case has to be framed by what the contentions, what the contentions are.

If the Court thinks they're fine to just have them the way they are, then the discovery will be relatively unbounded and that's the way it's going to go forward and we'll have to live with that. If the Court concludes that the things that we're asking for clarification on are appropriate, then that will give framework for the discovery. That will answer the discovery question as to what to pursue.

Just as one example, they said for PDF they think a standard PDF uses bitmap reuse because of the standards, right? I mean, it's more than just someone at O'Neil not saying it. It's in interrogatories, it's in HP and HP's witnesses. Everyone supports this. They're going to Global Graphics next week, which is the vendor of the RIP, and I think when they go there they're going to find out that there's no bitmap reuse in PDF. They can confirm that with a live witness. They may say they need documents, they don't have documents. But if the witness from the third party confirms what everyone else has confirmed, that there's no bitmap reuse whatsoever in standard PDF, all of this is a waste to pursue standard PDF as the infringement theory. I don't -- you know, I don't mean to beat a dead horse, but that's it.

But if they have a theory that standard PDF infringes, then they should set that out in a claim chart, if they think they have a Rule 11 basis, and then they'll -- they can have discovery on it. But until they set that out, it doesn't make a lot of sense to have mass discovery based on just the use of PDFs in all of these presses with variable data. But -- so I just -- it won't take a lot of time. They'll have done the deposition of Global Graphics. I think that Counsel admitted that that's going to be significant. Because the only piece they don't really have to verify, from my view, is the Global Graphics also saying when you have standard PDFs come through there's not reuse unless someone's setting it up. And HP said there is none, we're not setting any up, and the original user says we make all of the variable datas into individual documents beforehand so we don't believe there's any reuse. It'll be unanimous at that point in time, and I think that itself will be an important data point.

But if you can consider our motion, too, with that, then we can have an omnibus hearing in two weeks or whatever and then we can resolve all this, and if they're allowed to go forward on the contentions as they are, then they are.

THE COURT: Ms. Richards, have you seen this motion? Have you had a chance to review it?

MS. RICHARDS: We received the motion at approximately midnight last night, Your Honor.

THE COURT: That's when we got it. 11:30. But have you had a chance to look at it?

MS. RICHARDS: I had -- I briefly looked at it last night.

THE COURT: Is it going to be helpful to limit things?

MS. RICHARDS: No, Your Honor. From our perspective, --

THE COURT: I mean, is it worth sitting down with opposing counsel and discussing to see if you can resolve this on your own?

MS. RICHARDS: Well, I've spent about seven hours speaking with opposing counsel over the past two months and nothing seems to be moving. From IPT's perspective, these Defendants --

THE COURT: So, no?

MS. RICHARDS: -- are not going to do anything --

THE COURT: Okay.

MS. RICHARDS: -- until they get a court order.

THE COURT: What I'll do, then, is go ahead and look at the motion that was filed last night, even though it's not referred to me, in case it is referred to me. But I'll take a look at it and see if it helps me make a decision on this case. But right now I'm leaning to granting the Plaintiff's emergency motion and ordering the Defendant to provide these custodians. But I'll look at this and see if I can limit it as well. Now, we can --

MS. RICHARDS: May I --

THE COURT: We can meet again next week. My trial next week settled, so I have time Tuesday.

MS. RICHARDS: May I say one more thing, Your Honor, that I'm not sure came through? These preliminary topics are nothing more than preliminary topics. The way the ESI order works is they're supposed to identify their custodians and what they might have emails about, and then we get to make a small, narrow request --

THE COURT: Right.

MS. RICHARDS: -- with specific search terms and custodians. We've never gotten to that point --

THE COURT: Right.

MS. RICHARDS: -- because we have some information from O'Neil from their deposition and we just have one or two people and we weren't able to do that. So this idea that we're going to come back with something crazy and unbounded, we're not even allowed to.

THE COURT: I understand.

MS. RICHARDS: Thank you, Your Honor.

MR. REINES: Your Honor, I have a question.

THE COURT: Yes.

MR. REINES: My concern about this is actually to join things up earlier rather than later.

THE COURT: I understand.

MR. REINES: Because what's going to happen is they're going to have search terms and requests that I -- maybe I'm tilting at windmills, but that will be similarly unbounded, and we're going to have the same debate as to whether they have to have a boundary to what they're doing other than variable data printing. So that's what my concern is.

I understand the Court may grant it, and I guess it's unclear what the relief is, because under the rules it's 15 custodians. You know, we could get, you know, the hands-on printer people and whatever it is to do that. I'm not sure that makes sense, given the nature of those workers. But, you know, even with respect to Plaintiffs, they named like six people. So they didn't even comply with the 15. Because, frankly, I have to confess, in the context of their -- all these businesses, naming 15 is -- there's just not that many people that are likely to have real --

THE COURT: Right.

MR. REINES: -- evidence.

THE COURT: So you name all you've got. And if it's eight or six, that's what you name.

MR. REINES: But if it's all that I've got that's on variable data printing, then it's 150.

THE COURT: Yes. Understood. All right. Well, I'll take a look at it. And do you want to come back in? Do you want me to order you back in? What do you want to do? Ms. Richards?

MS. RICHARDS: I'm sorry, Your Honor. What will be the topic that we'll be discussing?

THE COURT: I don't know. Anything that's left out there to resolve.

MS. RICHARDS: We're always happen to come back in, Your Honor.

THE COURT: All right. Well, if it's not necessary, I don't need to waste your time. Are you local, all of you?

MS. RICHARDS: We are from Chicago and Denver, Your Honor.

MR. REINES: We're from -- I'm from California.

MS. MANESS: Houston, Your Honor.

THE COURT: Which explains the tan. And I think you're local.

MR. SCHROER: I mean, I am, Your Honor.

THE COURT: I just got my White Sox tickets last night, so I'm all set to go.

MS. RICHARDS: You can come and visit us, Your Honor.

THE COURT: Well, no, I got Ranger tickets with the White Sox.

All right. Let me figure out what to do. And I won't bring you back unless I feel it's necessary. We can just do phone calls since you're all out of town. I'm not going to drag you back here.

MR. REINES: I mean, if there's an important-enough thing, --

THE COURT: Well, I understand.

MR. REINES: -- I think, you know, you deserve that, but --

THE COURT: Well, you know, it's hard to travel.

MR. REINES: Understood.

THE COURT: It's not fun to travel anymore.

MR. REINES: I appreciate that.

THE COURT: Airports are not fun.

MS. RICHARDS: Thank you, Your Honor.

THE COURT: All right. Thank you.

MS. RICHARDS: Did you also want to hear argument on the --

THE COURT: No, I've got enough on that. I think, with this, I've got enough. But thank you.

MS. RICHARDS: Thank you, Your Honor.

THE COURT: All right.

MS. RICHARDS: Oh, Your Honor, will you also be ruling on the issue of the case schedule? Because that's very important to my --

THE COURT: I'm sorry, the what?

MS. RICHARDS: The issue of the case schedule. It's very important to my client. The fact discovery closes in June. And if it really is going to close in June, then we have no choice but to start taking these depositions even without documents.

THE COURT: Understood. No, I'll take a look at that as well and I'll try to talk to Judge Lynn about it.

MS. RICHARDS: Thank you, Your Honor.

THE COURT: All right. Thank you. All right. We stand adjourned. Thank you.

THE CLERK: All rise.

(Proceedings concluded at 2:10 p.m.)

--oOo--

CERTIFICATE

I certify that the foregoing is a correct transcript from the digital sound recording of the proceedings in the above-entitled matter.

/s/ Kathy Rehling **10/30/2015**

Kathy Rehling, CET**D-444
Certified Electronic Court Transcriber

Date

INDEX

IN THE UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

INDUSTRIAL PRINT TECHNOLOGIES, LLC, Plaintiff, v. CENVEO, INC. and HEWLETT-PACKARD COMPANY	CASE NO. 3:15-MD-2614-M Case No. 3:15-cv-00165-M
O'NEIL DATA SYSTEMS, INC. and HEWLETT-PACKARD COMPANY	Case No. 3:15-cv-01100-M
O'NEIL DATA SYSTEMS, INC. and HEWLETT-PACKARD COMPANY	Case No. 3:15-cv-01101-M
QUAD/GRAPHICS, INC. and HEWLETT-PACKARD COMPANY	Case No. 3:15-cv-01103-M
O'NEIL DATA SYSTEMS, INC. and HEWLETT-PACKARD COMPANY	Case No. 3:15-cv-01104-M
VISTAPRINT U.S.A., INC. and HEWLETT-PACKARD COMPANY	Case No. 3:15-cv-01106-M
FORT DEARBORN COMPANY and HEWLETT-PACKARD COMANY, Defendants.	Case No. 3:15-cv-01195-M

CONFIDENTIAL

TRANSCRIPT OF MOTIONS HEARING
BEFORE THE HONORABLE BARBARA M. G. LYNN
TUESDAY, JUNE 7, 2016
DALLAS, TEXAS

APPEARANCES:

FOR THE PLAINTIFF:

MR. STEVE SCHROER
FITCH, EVEN, TABIN & FLANNERY, LLP
120 SOUTH LASALLE STREET
SUITE 160
CHICAGO, ILLINOIS 60603
(312) 577-7000
scschr@fitcheven.com

MS. ALISON RICHARDS
FITCH, EVEN, TABIN & FLANNERY, LLP
120 SOUTH LASALLE STREET
SUITE 160
CHICAGO, ILLINOIS 60603
(312) 577-7000
arichards@fitcheven.com

FOR THE DEFENDANTS:

MR. EDWARD R. REINES
WEIL, GOTSHAL & MANGES, LLP
201 REDWOOD SHORES PARKWAY
REDWOOD SHORES, CALIFORNIA 94065
(650) 803-3022
edward.reines@weil.com

MS. AUDREY MANESS
WEIL, GOTSHAL & MANGES, LLP
700 LOUISIANA
SUITE 1700
HOUSTON, TEXAS 77002
(713) 546-5000
audrey.maness@weil.com

MR. M. BRETT JOHNSON
FISH & RICHARDSON
1717 MAIN STREET
SUITE 500
DALLAS, TEXAS 75201
(214) 292-4007
johnson@fr.com

PROCEEDINGS REPORTED BY MECHANICAL STENOGRAPHY, TRANSCRIPT PRODUCED BY COMPUTER-AIDED TRANSCRIPTION.

____

D. KEITH JOHNSON, RDR, CRR
FEDERAL OFFICIAL COURT REPORTER
FOR THE HON. BARBARA M. G. LYNN
UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF TEXAS
1100 COMMERCE STREET, ROOM 1572
DALLAS, TEXAS 75242
(214) 753-2325
keith_johnson@txnd.uscourts.gov

PROCEEDINGS

(June 7, 2016)

THE COURT: All right. So Judge Stickney is here in case, at the very beginning, anything that comes up that requires us to have Judge Stickney's good and wise counsel. Otherwise, he's got other matters to attend to. So whenever he wishes, he can just exit the scene.

So the Court has scheduled a hearing on various discovery matters on the IPT MDL.

I haven't seen you-all in a while, so I will cover the motions and anything else that is germane to the schedule as we proceed.

I will say as an overview of what I'm thinking about these, so we don't spend more time on them than we need to, that there is certain information that should be provided by each of you to the other. I am very disinclined to strike infringement contentions or to sanction anybody for anything that has happened here. And I am inclined to slightly further extend the discovery deadline.

I don't regard the matters that are before me as terribly dramatic or of crucial importance to the conclusion of the matter, as far as I am concerned. That's not a definitive ruling, but you-all should know where I am preliminarily coming out on this so we will attend to it appropriately.

All right. So I think we'll start with the plaintiff. I know that you-all have various matters, but we'll start there.

MS. RICHARDS: Thank you, Your Honor. Alison Richards on behalf of the plaintiff.

Would it please the Court for me to argue -- we have before the Court today our motion to compel and their motion to strike.

THE COURT: Well, they're going to get to start with their motions, and you'll get to start with your motions. So we'll start with your motion to compel, they'll respond, you'll reply, and then we'll cover their motions.

MS. RICHARDS: Thank you, Your Honor. Would it please the Court for me to argue from here or --

THE COURT: That's fine. As long as we can hear you, that's fine.

MS. RICHARDS: The record has become somewhat voluminous, so we created slides hoping to assist the Court in the efficiency of the argument today.

I believe that I'm plugged in. I'm not used to the Court's system.

THE COURT: Okay. There's limited -- a limited amount that I can do to help you. That should be coming up on my screen right now, but it isn't. I don't know what the problem is.

You-all should be trained in our system before you try to present something, because I am incapable of training you. So I don't know what the problem is.

MR. JOHNSON: Your Honor, may I approach the --

THE COURT: Yes.

(Pause in proceedings.)

MS. RICHARDS: Thank you, Your Honor. Alison Richards for plaintiff on our motion to compel.

So, Your Honor, just to begin, to make sure we're all oriented to the case, there are two patent families that remain in the case. On the left-hand side, we're talking about the inkjet sync patent. That's the '106 patent. As the Court is very much aware from the claim construction proceedings, there are both system and method claims at issue related to that patent.

The product -- the HP product at issue in that case is the HP inkjet machines, which HP also calls the Pagewide Presses or PWP presses.

What we've learned so far in discovery is that between the sales of the machines and the sales of the consumables related to the machines, we're talking about more than $400 million in revenue and $85 million in profits to HP.

The other side of the case is our five variable data patents. At this point in the case, it's the HP Indigo presses, a different press family that's at issue there.

Those are all method claims, which has had an impact on the how the case is litigated.

And what we've learned so far -- at least what we think we understand is that there have been about 2,000 Indigo presses sold in the U. S., each for about half a million or a million dollars. That math turns out to be between one and two billion dollars in revenue. We've also learned that related to those machines, HP sold a little more than a billion dollars in consumables, about $700 million in profit.

So on the variable data side of the case, things have gotten a little more complicated because of the method claims. And we have three different sets of infringers.

First, HP itself is a direct infringer. It prints variable data documents on its own Indigo machines. It does it at demonstrates; it does in-house printing; it does it to train customers; it does it to sell machines.

Secondly, we have the five printer defendants in the case, all of whom are also direct infringers. We believe that they're using the method claims.

And third, we have the claim that HP's inducing all of its customers to use these methods. That claim is not limited to the five printer defendants.

THE COURT: Is not limited to what?

MS. RICHARDS: The five printer defendants.

I don't think that there's any dispute about that.

So what's happened so far in the case is that we've learned -- and I think we agree with HP that there are three things that the Indigo machines can do that basically infringe. One, when they process PDF documents, they use a software feature called Optimized PDF. Two, when they process PPML documents for variable data print jobs, that infringes. And, third, when they process JLYT print jobs for variable data, that infringes.

So it means that some of the time when the customer's using the machine, they infringe by practicing the methods; other of the time they don't. It's important, because it relates to the discovery that's bringing us here today.

So there are also two sections -- excuse me -- to our motion to compel.

The first section relates to HP's own use. This is HP's own practicing of the method claims. And what we're looking for in this portion of the motion are some business record documents and a knowledgeable 30(b)(6).

Back in August, we asked HP, RFP Number 8 -- that's what's on the screen now -- documents identifying the instances in which HP practices the variable data methods. And what I understand from the briefing is that --

THE COURT: Ms. Richards, I'm flattered that you think this is big enough for me to read, but it isn't. Can you make that a little bigger?

Yeah. Thanks.

MS. RICHARDS: Thank you, Your Honor.

What HP, I believe, is saying now in response to this request is -- what I've gathered from the briefing is that they've given us Indigo job reports and they've given us a certain spreadsheet and that that should be enough.

But the problem is, Your Honor, that those two things don't identify all the instances in which HP has infringed. They don't give us a way to know how much HP has infringed. They don't give us any way -- it's basically a damages question, Your Honor. We know that HP practices the infringing method sometimes. The question is how often and when and to what effect.

The problem is, Your Honor -- and I'll -- I will attempt to zoom in -- the problem is that their witnesses -- their own witnesses have identified specific documents that exist that they haven't yet provided to us. And I'm specifically referring to the deposition of a witness called Ms. Matioli. This is taken from our reply, Your Honor.

THE COURT: Okay. Let me stop you for just a second.

Are you -- are you moving to compel on Request for Production of Documents Number 8?

MS. RICHARDS: I am, Your Honor.

CONFIDENTIAL - ATTORNEY'S EYES ONLY

THE COURT: Did I just miss that in your motion?

MS. RICHARDS: Yes. I believe it's in there, Your Honor.

THE COURT: Where is it?

(Pause in proceedings.)

MS. RICHARDS: It's definitely in the reply at Page 3.

THE COURT: Well, that won't get it. If you're moving to compel, it has to be in your motion, not in your reply to their response. I don't see it. I didn't see it and I don't see it.

MS. RICHARDS: In the opening brief on Page 4, Your Honor, at the top.

THE COURT: Okay.

MS. RICHARDS: Thank you, Your Honor.

So I believe that what HP is saying is they've produced the Indigo job reports and one spreadsheet, and they're saying that that's sufficient in the case. But the problem is that the Indigo job reports and the spreadsheet don't specifically identify the instances in which HP practices the variable data methods.

And their own witnesses, including Matioli, have specifically identified other documents that exist that do tell when HP practiced what we believe are the infringing methods. When did they print variable data jobs that are

Optimized PDF; when did they print variable data jobs that are PPML files; when did they print variable data jobs that are JLYT.

They have something, for example, called the Graphic Arts Experience Center, which apparently is a place where they have these very large Indigo machines where customers can go and customers can request, "Can we see how this print job prints and how fast?" HP can also show them, "Hey, we've got this functionality that works. Let us show you how this works," as part of selling this machine.

They also have something called the Graphics Solutions Center, which is a training facility where they show their own customers, here's how this functionality works; here's how your people can operate this machine to make this happen.

We also believe that HP visits customer sites where the Indigo machines are installed and puts on demonstrations. And the issue is, largely for damages purposes, when does this happen, how often does it happen, to what extent does it happen and those related things.

So we've not heard anything from HP yet other than the Indigo job reports and the spreadsheets are enough. And it's hard for us to understand how they can take the position they don't have to produce those documents when their own witnesses admitted that they exist.

Related to the same issue, discovery from HP about its own use, there's an issue related to a 30(b)(6) topic. The topic here was the nature and extent of HP's use of the accused variable data printing methods. The deposition occurred several months ago, and Ms. Matioli answered in response to seven or eight or nine questions on the same nature, "I don't know."

The questions were, you know, has HP used Optimized PDF, when, where; has HP printed PPML files that are variable data. "I don't know"; "I don't know"; "I don't know," which, you know, can be fine. That does happen at depositions sometimes. But it's been -- I believe at this point five or six months, we've asking for a witness that wasn't knowledgeable about those topics (sic). And HP has been saying no. And we don't see that there's any reasonable basis for them to say no to that. It's obviously relevant to both HP's infringement and damages. So that's the HP side of the motion.

There's another half to the motion, which relates to our inducement claim. We're at an interesting procedural situation, Your Honor, because one of the -- one of the issues that I wanted to ask you for relief on was RFP 1 today, but Judge Stickney's recent ruling has already compelled defendants to make a complete document production in response to RFP 1. So there is -- there is, unfortunately, some overlap that we didn't anticipate. But to make -- to make this part of the motion make sense, I think I'll tell you about it for background.

The issue is, Your Honor, that HP has all of these commercial printer customers that buy Indigo machines. And it's not just a one-off sale, like I walk into McDonald's and buy a Happy Meal, they didn't know who I was, they don't know if I'm going to eat it myself or give it to my kid. These are long-term, ongoing relationships that HP has with its Indigo customers. They're very expensive machines. As a result, HP and its customers create all of these documents about how the customers use the machines.

This, again, Your Honor, it's more of a -- more in the nature of a damages problem than anything. We know that there are Indigo customers that use what we believe is the infringing functionality. The question is who are they and to what extent do they use the infringing functionality. So again HP's own deponents have told us about documents that exist that would reflect what customers are infringing, but they haven't given us those documents despite our requests.

THE COURT: Okay. Is this something different than Judge Stickney's June 3rd ruling? And if so, what?

MS. RICHARDS: Well, I suppose that is in the eyes of the beholder, Your Honor. He compelled them to make a complete production in response to RFP 1. So I guess our view is that that issue has been resolved. Although this specific issue about the documents that HP identified was not before him, because this deposition testimony came out after that motion was filed and argued.

THE COURT: Okay. Well, I'm going to rule on what he hasn't. What he has, has already been ruled on.

MS. RICHARDS: I'll move on in that case, Your Honor.

The other issue, and the more important issue, Your Honor, relates to obtaining HP's knowledge about which of its customers infringe.

Back in March, we served a 30(b)(6) notice on HP. I'm sorry that it's so small. Let me tell you about it. It asks HP to tell us, to the extent of its knowledge, which of its customers use these infringing methods.

HP objected informally. They said we're definitely not giving you a witness on that. That's in the correspondence attached to the motion. They said could we do it by interrogatory? And we said, well, we're over the limit of interrogatories, and more importantly, we really don't want to wait 30 days to get an answer that we don't know or objections or citing back again to those same Indigo job reports.

So this original 30(b)(6) notice was never withdrawn. HP never provided any formal objections. They never gave us any authority that it was improper. They never filed for a protective order. They refused to give us a witness.

We sent them an interrogatory. By the way, at the bottom of the interrogatory, we said, "If perfectly complete information is not available to HP, please provide the information known or reasonably available to you."

So what happened is, they first agreed to answer on April 19th, and they sent over nothing, not even an explanation, "Sorry we're late." Just nothing. So we contacted them and said, okay, we'll do it by April 25th. April 25th came and went. No -- no correspondence, no "I'm sorry we're late," no "we're working on it."

So that 30(b)(6) notice has now been outstanding since March 22nd. They've blown through two deadlines when they said they'd give us the interrogatory response. We get the response, and of the 2,000, approximately, Indigo customers, they give us information about nine of them and say we don't know about the rest.

The problem here is, Your Honor, that's not the rule. Federal Rule 30(b)(6) says that HP has to give us the information known or reasonably available to HP. Rule 33 has a similar command, have to furnish the information available to the party. Got cases that say you've got to give us information within your control, what you know. Can't ignore information readily available to you.

So why do we think HP knows more? Well, the first thing is, their contracts with their customers entitle HP to the information that we're asking for. This is -- what's on the screen here, it's fuzzy on my screen. I'm not sure why. But in the record at A211, we put an example of one of these contracts. And in the contract, you can see HP requires its customer to allow them to monitor what they're using the press for, to collect data about what they're using the press for, to gather statistics about what they're using the press for. And then further in the same contract, the customers are required to maintain these operating connections at all times.

Same contract says HP's visiting these customers for up to four times per quarter. In the briefing, we've heard no response from HP about this whatsoever. They haven't denied that this means that the very information we're looking for is reasonably available to them.

Our view is that that alone should entitle us to relief. But there's quite a bit more. They've taken the position -- remember at the beginning of my presentation, I said there's three -- there's three pools of infringers: HP itself, the printer defendants, and all the customers that HP is inducing infringement.

As a defense in the case, HP has said that the printer defendants don't infringe. They don't use Optimized

PDF; they don't use PPML. So HP is saying we know those guys don't, as a defense. They had their 30(b)(6) provide a declaration after the motion compel to that effect. They provided an interrogatory response to the same effect. "No printer defendant enables the Optimized PDF setting." So when it helps them, Your Honor, avoid infringement, they have the information. But now we've asked, okay, are those printer defendants you're saying don't infringe, tell us about what your other customers do. And now they're saying they don't know. So we think that's a pretty big problem.

We took their deposition of one of the printer defendants, O'Neil, and we asked them about their use of what we believe are the infringing methods.

And my question was, "So how do you know that O'Neil doesn't reuse any elements?"

I'm sure you remember the extended discussion about reuse of elements from the Markman hearing.

"How do you know?"

And he said, "Through my experience and discussions with HP, HP would have told us."

So O'Neil is relying on HP in that situation to say they don't. But now we ask HP, okay, who does, they don't -- they're saying they don't know, despite the contract and despite these documents that exist that we don't have.

The O'Neil witness also described an extremely robust relationship with HP where they have hundreds of phone calls, fifty meetings. They're nearly in constant contact. HP definitely knows how these customers are using the Indigo machines.

We also took the deposition of a woman who works for Global Graphics. That was the provider of the RIP component in the Indigo machines. She told us that HP has an entire frontline -- that's what she called them -- of technical support employees that work directly with customers at all times. And there's a second line of customer support, people at HP who contact her at Global Graphics when things with the RIP go wrong. And she identified I don't know people -- there on the screen -- one, two three four -- eleven.

So this idea that we have no idea how our customers are using the machine, it just doesn't match up with everything else that we know.

But, Your Honor, there's more. HP's public documents, these quotes come from a gentleman named Alon Gazit, who works for Indigo in Israel, and he participated in some publications in which he says -- here what's he says: "HP Indigo has a large number of customers using PPML for their VDP jobs." I don't think that that's possibly reconcilable with the idea that HP has no idea what Indigo customers are using PPML for their VDP jobs.

He said something else. He said, "With HP Indigo VDP, our customers are experiencing double digit annual growth, and this new format will help them meet their needs."

He knows who these people are. And now we've deposed him. And you know what happened, Your Honor? This is tiny and very unclear. But he identified customers that practice the infringing methods that are not in the interrogatory response. There are nine customers identified and discussed in HP's interrogatory response for which they say this is the extent of our knowledge.

HP is large, and I can't take the deposition of every person that might know, nor do I want to, nor do they probably want me to. That's the whole point of Rule 30(b)(6) and Rule 33, is that they give us the information to the extent of their corporate knowledge.

This one gentleman identified five or six large customers. The question -- I apologize for my slide-making skills, Your Honor.

"Do you have a customer that you believe is practicing PPML for VDP?"

And there's an objection.

And he says, "Well, one of them might be Sandy Alexander, maybe Taylor." And then it goes on this way with him identifying customers that aren't in the interrogatory response.

So we think that HP should be compelled to either answer the interrogatory response or provide a 30(b)(6) with all of the information known or reasonably available to them.

THE COURT: All right. Thank you.

Response.

MS. MANESS: Thank you, Your Honor.

Audrey Maness on behalf of defendants. Just bear with me for a moment while I get connected here.

(Pause in proceedings.)

MS. MANESS: Thank you, Your Honor. May it please the Court.

I know Ms. Richards covered some of the overview of the case. And you probably remember quite a bit of it from the claim construction ruling.

As Ms. Richards mentioned, there's two press families involved here and two patent families with six defendants, HP and five of its customers. And all of the five customers are print shops that perform printing as part of their business.

I think it's important to note that Ms. Richards' presentation really established why determining what the scope of the case is so important.

Originally, IPT accused HP and its customers of infringing the -- the variable data patents whenever a variable data job is processed. That is no longer the case.

As Ms. Richards said at the beginning of her presentation, IPT is focused on three very specific types of variable data printing. One is the use of this optional Optimized PDF feature that is available on the Indigo presses but not on the PWP presses. That feature comes with the presses default off and must be turned on by a customer.

PDF is by far and away the most popular file type used. And at least what has borne out in discovery with the five defendants here is PDF is indeed very popular and none of these customers turn on that optional feature.

The other two file types that Ms. Richards mentioned were PPML --

THE COURT: When you say "none of these customers," do you mean the printer defendants?

MS. MANESS: The five printer defendants. Thank you, Your Honor.

And perhaps it would be useful for me to demonstrate with a graphic here of what processing a PDF file looks like. So here's an example. This could be a PDF job. It has three pages. It's a form letter addressed to three different recipients, and then there's a logo there in the corner.

When a PDF file is prepared by a designer at one of these print companies, everything is fully incorporated. If you're going to have a 100-page print job, you're going to send a 100-page PDF to the press. It will have the graphics on there. It will have the font and the text that you want. It will have what we've been calling the variable data, Dear Jack, Dear Susan, Dear Richard.

Generally -- and this is always the case with PWP presses -- the press will take each page of this file and rasterize -- that is, turn the image into print data -- rasterize the entire page, without regard to whether items are reused. So you'll see that the logo appears on every page. It doesn't matter to the PWP process. It will rasterize that logo every time it appears. There's no reuse in those systems.

And I think IPT has acknowledged that, and we can talk about that in more detail. But IPT has agreed in email that it will be dropping those presses from the variable data part of the case.

Indigo also processes PDFs in a similar manner. When Optimized PDF is turned off, again which it is by default, the Indigo presses will take the entire -- each entire page and create a bitmap, or raster image, and then send it off to the press to be printed.

PPML and JLYT are slightly different. And this is one of the reasons why they've become much less popular and they're more exotic file types now, is because when a designer is preparing a PPML or JLYT file, they're really preparing a set of instructions for the press. And it's harder to view, what's my final product going to look like.

So here you have an example of a file. And it's a set of instructions. Line 1, we're going it use Times New Roman font, we're going to put in the address for our company. And importantly, the PPML file can include assets within it, so the "my company" logo could be included within the file itself, or it could be referenced outside.

So it's a little bit of a different type of processing. And let me switch over here quickly to one of our other slides.

So I think this background is important to view IPT's most recent motion to compel. IPT has, as they noted, taken several depositions. We've been in fact discovery now for well over a year. And here's a timeline giving you a trajectory of what's happened.

And most recently, IPT, when it realized in February and March of this year that the printer defendants don't perform any of these particular, very specific methods of processing files, IPT turned its attention to, well, what are HP's other customers doing, what are HP's customers that are not defendants in the case doing.

And so it served as -- as Ms. Richards mentioned, it served a 30(b)(6) notice in late March. We had a discussion over the phone, and both parties agreed that that would be better -- more appropriate in an interrogatory. There was some correspondence back and forth. We were working very hard to gather the information. And indeed we did provide a response. This is giving you an overview of the discovery background here.

So we provided a response 30 days after -- after the deadline. And IPT has told you they're not satisfied with that, so they went and took a series of depositions.

I'd like to talk about that interrogatory, because it's very important. It asks for three things. First, it asks for a list of every HP customer that HP has sold to. And this is for Indigo presses specifically. We provided them that list. And I don't think that's in dispute here.

Second, it asked for an identification of the file types used by HP's customers. And this is a major sticking point here, is we referred them to the job origin reports. And Ms. Richards pointed to a sample contract that HP has with the Indigo customers. And the presses do indeed -- they're connected to the internet, and they report certain limited information back to HP. That information is contained in job origin reports.

The job origin reports are not fully comprehensive. Some countries don't allow for reporting back, so we don't have data from China. Some customers will disconnect their presses, even though they're not supposed to. But the job origin reports certainly are the most comprehensive source.

So IPT has said that no, these reports don't show where customers use PPML or JLYT. And Mr. Gazit, in his deposition, noted that these -- there are five customers that could be using these more exotic file formats.

Well, if you go -- this is an excerpt from one of our job origin reports that has been produced to IPT in the case almost a year ago. If you go to the job origin report, sure enough here you can see the data. This is from May 2015. You can see that Jeppesen is using PPML. Notably, Jeppesen uses PPML in a non-infringing manner or could even be --

THE COURT: Okay. How would I know that by looking at that? Is "Jeppesen Sanderson" on this document you're showing me in effect the title, so everything below it relates to --

MS. MANESS: Yes. And the formatting was off here. But the null and PPML files should be indented slightly, so those would fall under the Jeppesen entry.

Jeppesen printed some 3.5 million pages in May of 2015. Some of them were not able to be read by the press. That happens. But the ones that were able to be read, nearly 90 percent of them are PPML.

THE COURT: Okay. Just step aside for a moment. Stay there for a minute.

All right. Ms. Richards, let's just take this for an example.

Do you understand -- just confirm for me that this is an example of the job origin reports that have been provided and that you understood them or now understand them to be providing the information that was just stated?

MS. RICHARDS: This is not an example of the job origin reports that have been produced. This is apparently a blow-up of some fields that may be in a job origin report, and it reinforces the point that this alone doesn't have the information we need.

THE COURT: Okay. That -- that's beyond what I'm asking.

All right. Have a seat. I'll have you come back.

So I'm just going to regurgitate what I understood was just said to me by you, Ms. Maness, that -- first of all, is this the job origin report that was produced?

MS. MANESS: Yes, Your Honor. It is an excerpt -- it is an excerpt from a job origin report.

THE COURT: Okay. But it hasn't been reformatted; it is what you actually produced?

MS. MANESS: Yes, Your Honor. We produced them as native Excel files, and so fields can be turned on and off, and this was in one of the tabs, I think, as produced in the case.

THE COURT: Okay. The issue of whether something was or was not produced should not be the subject of a dispute before me, Judge Stickney or anybody else. So I don't know why that would be disputed. We'll come back to that.

Assuming that this is, in fact, what was produced, this says that of 3,504,852 pages, 3,261,412 were PPML.

MS. MANESS: Correct.

THE COURT: I don't know why there's this duplication of the same thing three times, but there is. They're not cumulative. They're just three saying the same thing three times. Three nulls, three PPMLs, all of the same number, not intended to be added.

MS. MANESS: Yes.

THE COURT: Okay. And I will do the quick math in my head that 3,261,412 and 243,440 is 3,504,852.

MS. MANESS: Yes, Your Honor.

THE COURT: Okay. All right. Go ahead.

MS. MANESS: Thank you.

And I've provided another example of one of the customers that IPT has complained about, and that's Taylor. Again, you'll see here that Taylor's reporting is largely PDF.

I will say, Your Honor, one of the fields that the job origin reports and indeed the Indigo presses do not report back is the use of that optional feature, Optimized PDF. And so that required some work on behalf of the attorneys and HP to gather what information it had about use of that feature.

THE COURT: Okay. So I'm going to take this in little pieces so I can digest them.

So step aside again for a minute.

I'm going to do this this way, because if I just wait for the whole thing, I'm not going to be able to get back to the level of detail that I think is required. So I'm going to have to have you-all attempt to enlighten me.

Let me ask you this question, Ms. Richards. If you, in fact, got this, just like it, with the exception of the Optimized PDF feature, does this document provide the information that you were asking for about the PPML and JLYT jobs?

MS. RICHARDS: No, Your Honor.

THE COURT: Why doesn't it?

MS. RICHARDS: Because knowing that it's a PPML job alone is not sufficient to determine if it's infringing.
Ms. Maness herself just said, okay, this one -- not the one on the screen.

THE COURT: Go back to the one that you had, Jeppesen.

MS. RICHARDS: Okay. So here's Jeppesen before the Court, and it says that there are PPML. That's a fine piece of data. But knowing that it's PPML alone is not sufficient. She just said in a voiceover that this one doesn't infringe, I guess, because it's not PPML for variable data processing.

So you can have a PPML file that infringes or a PPML file that doesn't infringe, which we've said in the case is a variable data file or is not. This alone says it's a PPML file. And that's fine, but I have no way of knowing, looking at this, if it's a variable data file or not. It could be just an ordinary file. That's the key fact.

THE COURT: Okay. All right.

Ms. Maness, back at you.

MS. MANESS: Thank you, Your Honor.

And as to the --

THE COURT: Excuse me just a minute.

(Pause in proceedings.)

THE COURT: Go ahead.

MS. MANESS: Thank you, Your Honor.

As to the specific content of what's included in every job printed by any one of these print shops, that's where HP's knowledge ends. We can tell you for May we have this report that most of the jobs are PPML. We know about Jeppesen, because they were asked in a recent deposition, and the HP witness said, "Well, last I know, Jeppesen was printing flight manuals. Flight manuals are not variable work." So there's some offhand, anecdotal knowledge about what types of jobs certain companies may be printing, but we do not have a complete record of that.

THE COURT: So I took Ms. Richards' comments to be but you could get it; pursuant to these contracts, you have the right to get it.

MS. MANESS: No, Your Honor. Those contracts establish -- provide for connectivity of the machine to Indigo headquarters.

THE COURT: Okay. All right.

MS. MANESS: But the data reported back is limited.

THE COURT: Okay. Step aside. Sorry. You're going to be getting cardio today.

All right. Ms. Richards, okay. I take your point that the information that you've asked for is not all here, even assuming that you got it. But the position of the defendants is, "We don't know, and what we get is what is produced to us sort of automatically about how it's being used by HP customers, except, footnote, where it's prohibited by law in another country or where somebody turns off that function contrary to the contract."

Okay. So what's your response to that?

MS. RICHARDS: Well, I have a right to get information, by contract, and I just choose to take Items 1, 2 and 3, but I don't take Items 4 and 5, still under the rules, Items 4 and 5 are still reasonably available. If you look under the contract, it doesn't say -- it's a very, very broad right. They can monitor data for any reason. They can

CONFIDENTIAL - ATTORNEY'S EYES ONLY

collect data for any reason. There's nothing limited there that says you can only have items of data 1, 2 and 3. So that's the first thing.

Second, they have no way of knowing who is using Optimized PDF, because they've gone out of their way to represent to us that means five printer defendants aren't using Optimized PDF. How can it be that HP knows that those five aren't using it, but come in and say, "I don't know anyone else is using it or not." That can't be. You either have access to the information or you don't.

Your Honor, to be clear, I don't want to leave you with the impression we didn't receive the Indigo job origin reports. We did. And we're not taking that position, you'll see in the record.

But I think what I was trying to convey is that counsel has altered the report here. And I don't mean that in a pejorative way at all. What we're looking at on the screen is perhaps an excerpt of these very large Excel files. I didn't want to leave you with the impression we were saying we didn't get the --

THE COURT: Well, I think you left me with the impression that what you got was something different from these. And I understood what you got was these, although you've got much more than these. Is that correct?

I'm not suggesting by that comment that you got

CONFIDENTIAL - ATTORNEY'S EYES ONLY

this detail that you're now requesting. But that this same data, not altered, the same data was included among what was, in fact, produced.

MS. RICHARDS: I have no way of knowing that for sure, but I don't dispute it.

I have an actual job origin report on my machine. It's a very, very large spreadsheet and -- so I can't tell you.

THE COURT: Okay.

MS. RICHARDS: And there's no point in arguing about it.

THE COURT: So are you -- let's take Jeppesen, for example. So is it your position -- let's assume for the sake of discussion -- and I don't have any reason to think this is not accurate -- that this is the information that's generated, sort of automatically, by the printers of Jeppesen to HP, subject to they turned it off on occasion, subject to there's a printer in China, those kinds of exceptions.

But with those exceptions, this is, in fact, the information that is automatically generated. It doesn't -- those reports do not provide the level of detail that you want by your interrogatories. But HP doesn't have the detail that you want. Your position is but they can get it and they should get it?

MS. RICHARDS: That's a part of my position. The other part of my position is the Indigo job reports are one source of the data bucket. If you read their filings carefully, they're not saying there is no other source of data. Two of their witnesses have already told you that there are several other sources of data.

They have these employees known as solutions architects who work every day with their customers, and they have produced one example of this form they filled out called -- I believe it's called PQD. It's in the record. And in that form, the solutions architect writes, "I'm working with" -- I can't recall who the customer is; I think it might be O'Neil -- and they're using PPML and printing this kind of a print job.

There's a whole universe of other sources of data. HP is saying, well, we gave you the job origin reports. And that's fine, but I don't think that they're saying we've conducted an exhaustive search and there is nothing else. At least that's not how they've ever responded to correspondence or in their briefing.

THE COURT: Okay. Back to you, Ms. Maness.

So let's talk about whether you -- you corrected me. I'm not offended. Just trying to get us back to where we were -- when I suggested that the I thought the plaintiff's argument was that you had the right to go get data, and you responded, no, the contracts provide connectivity; that is,

CONFIDENTIAL - ATTORNEY'S EYES ONLY

there will be this sort of automatic download or upload of data to HP, but that the contracts do not support the argument that HP can go knock on the Jeppesen's door and say, "I know we've gotten 100 percent of what you're supposed to send, and we have never turned it off, but we want to know more detail about what you were doing when you were using the PPML.

MS. MANESS: Yes, Your Honor. You've characterized it correctly. There's automatic reporting of -- of particular types of data, i.e.: "Are you using PPML? Are you using PDF? Are you using JLYT? How many pages did you print?" What is not automatically reported and what we are not entitled to go get, is knock on Jeppesen's door and say, "Tell me how many of those pages had variable data in them."

THE COURT: Okay. So what is the specific reference -- and I don't have it with me. Becca will get it for me.

Ms. Richards, what is the specific contract reference that you say supports the notion that they have to go get more? And then, Ms. Maness, after I cover this subject, then we're going to cover the other sources of material.

So, Ms. Richards, can you give me that reference, please, in the contracts?

MS. RICHARDS: Yes. It is in the record on -- it's an HP Supplies and Shared Maintenance Agreement. This is a sample. The relevant portion is at A211, which is also IPT HP 1521. And in the same document, there's another relevant section, Section 7, which is at IPT HP 1523. And we've cited a third page of the same document at 1522 that talks about HP visiting a customer up to four times per quarter.

THE COURT: Okay. So this question I'm about to ask, I just have no idea what the answer to this question is. So this is not a question I'm trying to confirm a suspicion; I don't have a suspicion.

So this particular report I'm looking at is May of 2015. Let's assume, Ms. Maness, that you did have the right to go get it, the information, just for the sake of this discussion. Is the kind of information that is being requested about the detail regarding this report, one year later, still accessible?

MS. MANESS: No, Your Honor, it would not be accessible on the DFEs themselves.

THE COURT: What is that?

MS. MANESS: Digital front-end, the press computer.

THE COURT: Right. I don't mean that. So --

MS. MANESS: Would it be accessible if we just asked?

THE COURT: Let me just back up to that.

Was that ever accessible on that computer? I thought not.

MS. MANESS: No, Your Honor. No.

THE COURT: Okay. So I'm asking a different question.

If -- if HP knocked on Jeppesen's door and said, "Okay, we got these reports generated by the computer, now we want more detail," is that kind of detail -- you may not be able to answer. Is that kind of detail retained by customers as far as you know a year after the job?

MS. MANESS: It's a complicated answer, Your Honor, because it depends on each print shop. And we've seen this with the printer defendants in this case.

So some printer defendants, it's a fairly easy answer. O'Neil Data Systems prints largely variable data documents. So they could say, yeah, it was probably a variable data document, from memory and based on our business.

Other printers have a mishmash of, you know, one shop will do big customized banners; another shop works on the Coca-Cola labels with the names. And asking for recollection one year back and then six years back is going to be inaccurate certainly.

THE COURT: Okay. And how many -- just give me ballpark, if you can. How many customers -- let me ask it a different way.

The data for how many customers is in these job reports? A thousand, a million, ten, what? How many?

MS. MANESS: It is in the low thousands. It's worldwide.

THE COURT: Okay.

MS. MANESS: So in the low thousands. Several customers have multiple presses. And in the U.S., we have I believe it's around 300 Indigo users.

THE COURT: Okay. And you're just going to have to remind me. I just don't remember. Is this claim limited to Indigo users in the United States?

MS. MANESS: Because it's a method claim --

THE COURT: Yes.

MS. MANESS: -- the use has to occur in the U.S., yes.

THE COURT: So there's roughly three -- is that a number you're comfortable with, Ms. Richards, that Indigo customers in the U.S., it's about 300?

MS. RICHARDS: Yes, Your Honor. I may have introduced some uncertainty into the discussion. The interrogatory doesn't ask how many pages and how many jobs. That's not information we're seeking. The only question is have they ever used this format.

So I'm not suggesting that Ms. Maness go to 300 customers and ask them about every job and every page. The question is only has the customer ever used it, which is the --

THE COURT: Well, I have to say, I'm a little -- if the only issue on these is what kind of printing do you do, why can't there be a deposition on written questions on these? That's a very simple three questions to 300 people. Why is that so complicated?

MS. RICHARDS: Well, that is our next option, Your Honor, which was -- we would need a little bit more time to make that happen.

But we first actually asked HP, hoping that it would be less burdensome, based on their own documents and own witnesses' testimony that they could access information. We've gotten some complaints from HP, Your Honor, that they think that we're trying to harass their customers. And we believe that sending discovery to each U.S. customer would be less -- would be more burdensome than just asking --

THE COURT: Okay. That -- I'm not now ruling that that's what should happen here. But I can argue both sides of that equation, if I were HP, whether I'd rather blame it on you or do it myself. I could argue both sides of that. I've been there with clients, and sometimes you elect to do the work yourself and keep the other guy out of it. Sometimes you like to blame it on the other guy.

So I don't know which is correct, but I hear you about the information -- I understand why the plaintiff wants the information. It seems quite clear from both of you that

HP doesn't have the information.

I don't hear you saying that HP doesn't have the information. You're saying HP can get the information.

MS. RICHARDS: I'm saying HP has the information in other documents.

THE COURT: We'll get to that. That's fair. We'll get to that. Thank you for reminding me of that. That's the next question I teed up, and we'll get to that. But for the moment, let me just put that to the side.

So on this question, if the only documents that HP has don't have this information, unless Ms. Maness could convince me it wasn't actually relevant, then I would pitch this question to HP saying, what do you prefer: Going and getting the information yourself, without me necessarily holding that you have the right to require it, or would you rather let them go get it if the format -- you don't have to answer that right now, Ms. Maness, because that would require Mr. H. and Mr. P. to decide.

MS. MANESS: Thank you, Your Honor.

THE COURT: Okay. But -- so I want to put that to the side for the moment.

Let's talk about the other documents. So Ms. Richards is suggesting that there's information out there that this information is, in fact, available. It may not be scientific, but that if I laid out a list of 300 customers and

CONFIDENTIAL - ATTORNEY'S EYES ONLY

you asked the right people, that there are people at HP who could, with knowledge, say these 10 customers do this kind of printing, these 20 do this kind of printing, these 30 do this kind of printing, under oath. Is that correct?

MS. MANESS: Your Honor, the information that they would provide would still be incomplete and anecdotal. And I think we've mentioned this in our motion, is that the solution architects that Ms. Richards mentioned, these are technical folks who do interface with customers and deal with customer problems to the extent that a customer has a problem. So they may find out -- they may have a picture or a snapshot into the work being performed at that particular time.

Again, customers can change what they do. A customer may call a solutions architect with a prospective question, "Hey, I'm thinking of switching to this other workflow. What do you think?" Well, do we know if they switched? No. We'd probably have to go and check -- I mean, switched to PDF from PPML. We'd go check the job origin report to see if they ever switched.

THE COURT: Well, but that -- what you just said, that's available in this --

MS. MANESS: Correct. What they are actually doing, the source of what they are actually doing is the job origin reports, or what they've actually done.

THE COURT: All right.

MS. MANESS: Let's --

THE COURT: I'll come back to where you are.

But let's take Jeppesen. So I don't know -- what's the last date of these that were produced?

MS. MANESS: I think we produced them through August of 2015, and we made the full production in October.

THE COURT: For the sake of discussion, that's fine.

So let's say -- and I don't know. What's the beginning date?

MS. MANESS: They started collecting this date in -- it was either in late 2011 or 2012.

THE COURT: So let's just assume for the sake of discussion that I have four years of monthly reports for Jeppesen. And for the first two years they're all JLYT, and then for the last two years they're PPML.

So that degree of change, that kind of change, is going to be on the job reports. The change in detail of the work that Ms. Richards is interested in is not on these reports, because that kind of data is not captured. But the kind of change between the processes is.

MS. MANESS: Yes, Your Honor. And I should add that I was the one that did most of the undertaking for this rog. response, and it involved interviews with nearly three dozen HP solution architects and project managers and everybody that I could find that might have knowledge, I talked to and said, "Is there a better source than the job origin reports?" And their response uniformly was, "No, we might know at one point in time and then find out a year-and-a-half later that they've switched, but you're going to need to go back to the data to check."

THE COURT: Okay. All right.

MS. MANESS: And so let's talk about the two other documents that Ms. Richards brought up. Those came up in a deposition of Scott Cazel in Chicago a couple of weeks ago.

One of them is a PQD. If a customer approaches HP and has a specialized request, you know, "We're thinking about overhauling our workflow; we're going to use this new front-end software; will the DFEs -- the Indigo DFEs that you have -- we already support it or do we need to upgrade?" HP will put together a PQD. From my understanding, this is not a common practice, because it's usually when customers -- when something's working for a customer, they're not going to change it. But when they do, they put together this PQD.

The one that Ms. Richards was referring to -- I believe it was for Seveyo (phonetics) -- they never ended up following through with that. So they're all a handful of PQDs. And if Your Honor would like, we can find them all and produce them. A lot of times they don't have to do a variable data at all, because Indigo presses, 90 percent of the time, are printing static data, not variable jobs. But they're not going to tell you what a customer is doing. They might tell you what a customer is interested in or was interested in in 2012. But whether the customer followed through is not going to be reflected in that document.

THE COURT: What does that stand for?

MS. MANESS: It's qualification Document, and I can't remember what the P is.

THE COURT: Okay.

MS. MANESS: And a similar document is this workflow analysis. And if I'm remembering correctly -- I don't have it on me. But again, that's a prospective document. Or if a customer is having problems with their current workflow, the technical team at HP may get together and say, "Okay, well, let's -- let's figure out what they're doing, where the problem may be, and maybe we can recommend a solution to them."

Again, it might give you a snapshot at that point in time of what the customer is doing generally. Oh, this customer is processing PDFs. It may not even tell you whether there's variable data in those PDFs or not. And from my understanding, it's certainly not going to get down to the level of what's the particular configuration on the press; are they selecting this optional feature.

But again, it's prospective looking. So the level of information provided is going to be incomplete at best and inaccurate at worst.

THE COURT: Okay. And what does this spreadsheet contain that the job origin reports don't?

MS. MANESS: Can you refer me to --

THE COURT: Is that limited to HP use, not the customers?

MS. MANESS: Correct, Your Honor. There's a spreadsheet that we've produced --

THE COURT: Okay.

MS. MANESS: -- that identifies all of the demos.

THE COURT: Okay. All right. I'm going to have you -- you have to come back and talk about the rest. But let's try to bring this issue to a closure of job origin reports.

MS. MANESS: Thank you, Your Honor.

THE COURT: You can lean -- you can lean or sit down as you like.

MS. MANESS: Thank you, Your Honor.

MS. RICHARDS: I actually have a PQD right here on my laptop. And it shows that this customer was using PPML, and there's a checkbox --

THE COURT: I'm going to get you all of those. I'm going to require HP to produce those. I think there's a handful of those. If you don't have them, you'll get them.

MS. RICHARDS: Thank you, Your Honor. I'll spare you the --

THE COURT: Okay. But I want to come to closure on this.

I'm not convinced, Ms. Richards, that -- I think you've got some examples of a particular person who, because of familiarity with particular customer, can speak with some degree of authority about their -- sort of the general M.O. of their business. But it seems quite obvious that that is not detailed data. And that is -- it's not a snapshot, because it will be good information as long as the person has that degree of a detailed relationship.

But if you really want the detail, I don't know how you can get it except to go get it from the customers. And if you get it from the customers, it seems there's two ways to get it. One, I deputize HP to go get it, or you go get it yourself through a pretty simple, seems like, three-question deposition on written questions.

I'm going to let you respond to that generally, Ms. Maness, without deciding, if those are your choices, what you want to do. But why doesn't that serve the purpose, one of those methods?

MS. RICHARDS: The two methods being --

THE COURT: They ask -- I -- what I'm reading of the contract, in my view, doesn't -- I don't read this the same way you do, that they get to knock on the door and get whatever they want just because of the contract. But that doesn't mean I would not give them the option to go get it if they convinced me that 300 depositions on written questions is going to unfairly burden the relationship.

But it would seem to me that you could get that data from the customer, either through HP's effort or through short depositions on written questions.

MS. RICHARDS: Your Honor, I'm here today because my client needs the data. So however I get the data, my client will be happy, Your Honor.

THE COURT: Okay. While I have you up -- I should have started with this. This is the time for us to agree to what extent the nature of this case has narrowed as time has passed.

So Ms. Maness began her presentation with you were claiming that there was infringement whenever a variable data job was being done, I think is the way Ms. Maness put it, and that's no longer your claim.

MS. RICHARDS: That was never our claim, Your Honor. I actually --

THE COURT: Okay. What is your claim -- what is your claim about?

Let me ask it this way: Have you narrowed any of your claims from what was originally asserted? And if so, in what way have you narrowed it?

MS. RICHARDS: We have, Your Honor.

So at the outset, we talked about the inkjet PWP side and the variable data side. The PWP printers are no longer accused on the variable data side. And the reason is because we got documents and we reviewed source code that confirmed what one of HP's witnesses told us.

Now, HP has been pushing us very, very hard to limit our case on the variable data side based on their attorneys' representations that those printer defendants do not infringe. And for quite some time we've had a dispute where we've been saying, look, this is not a plaintiff that wants to maintain meritless claims. That's not what we're about. We've already dropped some claims. But we're not going to drop claims and agree that someone doesn't infringe just based on carefully crafted attorney statements.

The orders we got from Judge Stickney Friday and again yesterday address that. We've got the documents coming; we've got the testimony coming. If the documents show that they're not infringing, we will fully address that. If the documents show they are infringing, we're very much entitled to maintain those claims.

So there may be more -- there may be more grooming coming, Your Honor, but certainly not until we get that discovery that was ordered.

CONFIDENTIAL - ATTORNEY'S EYES ONLY

THE COURT: So have the HP Pagewide Presses, that are referred to in the briefing, as the PWP presses, are those still in the case?

MS. RICHARDS: They're on the inkjet side of the case, Your Honor.

THE COURT: Yeah. Are they still in the case?

MS. RICHARDS: Yes.

THE COURT: So they don't think so. So why do you not think so?

MS. MANESS: I'm sorry, Your Honor. If we could clarify. The two inkjet patents -- now one -- is asserted against the PWP presses only.

So now we have -- before it was variable data patents against both Indigo and PWP. Then inkjet was only against PWP. Now, based on my understanding of what Ms. Richards is saying, it's going to be variable data against Indigo, period; inkjet against PWP.

THE COURT: Right?

MS. RICHARDS: Yes.

THE COURT: Okay. I misunderstood what you were saying, Ms. Maness. All right. That's helpful.

Okay. Thank you. You may be seated, Ms. Richards. Thank you.

All right. So, Ms. Maness, before we move on to another subject, let me hear from you on this issue of you go get it. I'm not asking you to pick. But why should I not either -- and I recognize the discovery deadline here. The Court can modify the discovery deadline. The discovery deadline has been modified to a degree. If I grant your motion, I'm likely to modify the discovery deadline a bit anyway.

So why would I not say either HP go get it or let the plaintiff go get it?

MS. MANESS: Your Honor, again, I'd like to check with my client. But I think the appropriate course of action is who has the best source of information of what's being printed. The customers who are printing it have the best source of information.

THE COURT: Well, I don't disagree with that.

This option, which is frankly somewhat cumbersome for HP, because HP would, in effect, have to vouch for the customer information. That's tricky. But sometimes people like to go contact their own customers. But as I said, sometimes they like to blame it on the third party. So I will leave that to you.

Okay. All right. So I think on that subject, I think that the information -- I'm not convinced that HP has not provided the information it has. I think that plaintiff is entitled to get that information. And it can either get that information through simple depositions on written questions. The Court would -- assuming that the questions are what I anticipate they would be, I will accelerate the date on that. It's going to be subject to HP getting back to the Court by Friday.

Is that reasonable?

MS. MANESS: Yes, Your Honor.

THE COURT: To say -- I'll say pick your poison, but I know you will present it in a much more uplifting way.

MS. MANESS: Thank you, Your Honor.

THE COURT: Okay. All right.

MS. MANESS: May I address the scope issue real quick that Ms. Richards was talking about?

THE COURT: Yes.

MS. MANESS: And I think this is going to go towards the infringement contention motion in general. But from what we understood, the original infringement contentions to be was all variable data -- Indigo supports PDF, PPML and JLYT files. Anytime those files include variable data, they infringe. That's what we understood -- and the contentions still say that.

THE COURT: Let me short circuit this for a minute.

MS. MANESS: Yes.

THE COURT: I said at the beginning, and I feel this way still, that I'm not going to strike the infringement contentions because of the issues raised in your motion to compel. But I am going to require that the infringement contentions accurately reflect what is claimed to infringe. And they don't.

MS. MANESS: And, Your Honor, to that point as well, Ms. Richards was saying that we shouldn't be forced to accept attorney representations on what file types the printer defendants are using. It's not just attorney representations. It is interrogatory -- detailed interrogatory responses; it's 30(b)(6) testimony in the case of O'Neil; and to the extent Your Honor is granting relief on the infringement contentions --

THE COURT: Okay. Well, let me -- I'm going to interrupt you, because I know where you're going.

MS. MANESS: Okay.

THE COURT: I'm not going to at this point anticipate what might or might not happen as far as the plaintiffs are concerned after the information that Judge Stickney ordered is produced. At that point, I believe you're going to be arguing that certain claims should fall away. And if they don't, you presumably will be coming back to me and saying, this is not just Audrey Maness telling you this; this is -- in twelve different ways, we've communicated in an enforceable -- enforceable is not the right word -- in a legally cognizable manner that the facts are X, and one cannot in good faith pursue a claim on that. I don't know how this is going to come out, Ms. Richards. I didn't mean my comment to anticipate that. But that opportunity is still available to you. I'm just not ruling on that now, because they haven't said that they're not going to marry their claims. Let's let this information come out the way it's been ordered, and if your position is that they can no longer proceed in good faith on the printer defendants, then you can file appropriate motions.

MR. REINES: Your Honor, I apologize for interloping here. But --

THE COURT: No, you don't. Not really.

MR. REINES: Well, obviously I am.

But the Court brought up the sequence of events in the remainder of the case, which is a broader issue. And, you know, respectfully, we've reviewed Judge Stickney's order, and we have, you know, fairly substantial concerns about it at a number of levels, not least among them what's now undisputed on the record here that they don't challenge standard PDF processing, which is 90, 99-whatever percent, depending on which print house it is, and they only challenge Optimized PDF and these two other species of formats.

Judge Stickney's order was based on the theory of the case expressed in the infringement contentions, which is why one primary basis for our opposition was we need the infringement contentions clarified so we're not giving discovery on just PDF use, which doesn't have any reuse, has nothing to do with the case.

And it would be a massive waste, and frankly a massive injustice, if we had to produce documents as though the entire O'Neil printing operation was being accused of infringement when none of the O'Neil printing operation is being accused.

THE COURT: Okay. I understand what you're saying.

MR. REINES: So we would like to --

THE COURT: Just a minute. I am not prepared to modify Judge Stickney's order. I understand what you're saying, and this is what I suggest about it.

In light of what you -- what the plaintiff has agreed is no longer in the case -- and can't tell as I sit here how that impacts, if at all, Judge Stickney's order.

If it is the position of the defendant that something that has been said today, if known to Judge Stickney, who is not here with us at the moment, would cause him to modify his order, the way I want this to be handled is the parties will confer. If you don't agree, as I anticipate you will not, then you may file promptly a motion to modify Judge Stickney's order, which will be based only on something on the record in the proceeding today. And you've got the record. If you think something has been said that should cause a modification, then you file a short motion to that effect.

If you otherwise disagree with Judge Stickney's order, then you will appeal it to me. You will not file a motion for reconsideration on his order.

MR. REINES: Understood. The only issue we have with respect to the -- the former point is very helpful. We'll follow that procedure. We could do that through an appeal to you or modification for him. If the preference is that we go for the modification, that's what we'll do, as to the admissions made today, which is what we've been asking for for months.

I mean, the rub of the whole dispute, all of the motions, is that we want the case limited to Optimized PDF rather than broadly any PDF, because there's no basis for any PDF. But now we finally have the admission on the record here that's not being backed away from.

MS. RICHARDS: I'm not certain that that's true at all.

THE COURT: Okay. I didn't hear this said in the way that you are arguing it was put. But I've gone as far with it as I'm going to. What has been said has been said. You have the record. You'll get the record, go back to Judge Stickney, if there's a basis for it, and that's all I'm going to do with it. I'm not otherwise modifying his order. And I'm directing that you may not move to modify his order. If you have an appeal from it, you will have it to me. The likelihood that I'm going to stay discovery that's been ordered by Judge Stickney is pretty remote.

MR. REINES: Yeah. That was the next issue I was going to raise.

THE COURT: Unlikely. You can ask for it. I'll rule on motions as I get them. Unlikely.

MS. RICHARDS: Your Honor, may I make a one-sentence statement to make sure that there's no unclarity?

We think that we've reached agreement with HP that Optimized PDF, PPML for VDP, and JLYT for VDP infringe. That doesn't mean -- so Optimized PDF is this name that HP has given for this one feature. I think everyone agrees that it infringes. But we've been saying for months -- O'Neil's attorneys have been saying we don't use Optimized PDF, therefore, we don't infringe.

We've been saying, well, just because call something Optimized PDF or not is not the dispositive issue of whether something infringes. It's whether it meets the claims. So show us how your processing works. And if it meets the claims, it meets the claims. If it doesn't meet the claims at all, then we'll be reasonable about that.

But I don't want there to be any unclarity on the record that something has to be called Optimized PDF to fall within our contentions. That was the whole point of our motion to compel before Judge Stickney is how do you do the processing. He ordered we're entitled to the technical documents that show how the processing is done, whether or not it's technically called quote, unquote, Optimized PDF.

MS. MANESS: Your Honor, if I may, slight clarification.

We don't agree on infringement. The question is what's in the case, what's accused. And so it sounds like the use of the Optimized PDF feature, to be clear, PPML, when it includes variable data, and JLYT when it includes variable data.

Is that correct?

MS. RICHARDS: Yes.

THE COURT: Yeah. That's --

MS. MANESS: And then there's the qualification of at least that. And that's sometimes creating the problem.

And if I may address the documents point, I think what Ms. Richards is and IPT have been asking for is documents that demonstrate O'Neil, for example, does not use the Optimized PDF function. Well, I can give her a screenshot of the user interface that shows it's not clicked. But as we said in our motion, that's like asking you, Your Honor, show me that you don't use a certain feature back in chambers. Well --

THE COURT: Well, I -- what I understood

Ms. Richards to be saying is that -- I'm not sure she concedes that Optimized PDF is necessarily a term of art that is satisfied by the function that can be turned on by the customer, that there could be something equivalent to that that is actually done by the customer.

And I think Ms. Richards is saying, though not capable of confirming or rebutting the notion that Judge Stickney understood that, so that what was ordered was not just a confirm that the customers do not actually turn on the Optimized PDF switch, but something more. Is that right?

MS. RICHARDS: Yes, Your Honor.

THE COURT: Okay. So that's as far as I'm going to go.

So I'm not sure that this -- I hear you, Ms. Maness. You're saying it's only cap O, cap P, cap D, cap F. There's a switch for it. We don't use it. It's gone. Ms. Richards is saying, that's not what I mean. I mean this function of optimized PDF doesn't have to be all caps.

I don't know what was presented before Judge Stickney. I just don't know. I wasn't at the hearing. So I'm not going to go any further than this. I think I teed this up specifically enough that if there's a motion that he can evaluate what it was that he intended. So I'm going to leave it there.

MS. MANESS: Understood, Your Honor. And just so that the record is clear, whether a customer formats a PDF in a certain way or tags certain things, ultimately all of the claims require that it be processed a certain way in the HP press. And IPT has the information on that, has source code, has detailed interrogatory responses; it has 300,000-plus pages of documents. I think that's where we're concerned of what more could you possibly need.

THE COURT: Okay. Well, we're not redoing the hearing before Judge Stickney right now. We're not doing that.

MS. MANESS: Okay, Your Honor.

THE COURT: I've got enough before me, and I haven't made much progress. So I said how I'm going to approach that, and that's how I'm going to approach that.

MS. MANESS: Where would you like to go next? There's one more part to this interrogatory.

THE COURT: Okay. Go ahead with that.

MS. MANESS: So because I know you want to hear more about Optimized PDF, the last part of the interrogatory asks for every HP customer that HP knows.

THE COURT: That's when I -- before I went to bed last night, I said --

MS. MANESS: I love Optimized PDF. Now you'll dream about it tonight.

THE COURT: Not likely.

MS. MANESS: I hope not.

So IPT asked for HP's knowledge of customer use of this feature. And the facts are important. It's shipped -- these presses are shipped to the customers; this feature is default off. The customer can turn it on. This is not one of the datapoints that is reported back to HP.

So then the question is, well, how would HP know whether a customer is using it or not. And we explained to -- to IPT that we're going to have limited knowledge on this; we'll give you what we have. And as part of that, interviews -- again, interviews with nearly three dozen HP employees that have customer interactions or deal with customer technical issues. And what we found was nine customers that either used or tried to use the feature at some point. Usually HP found out about it through a customer escalation. So a customer called HP and said, "I'm having a problem using this feature." And all of that information is detailed in the interrogatory response.

THE COURT: Okay. Well, if you're either going to the customers or they're going to the customers with a deposition on written questions, they can put a question on this -- in the questions or you could.

MS. MANESS: They could, yes, Your Honor.

THE COURT: All right. Okay.

MS. MANESS: I think that wraps up -- well, would you like to discuss the Teresa Matioli -- HP's use is the final --

THE COURT: Yes. It's the last thing I want to talk about on this.

MS. MANESS: So --

THE COURT: So this is what the spreadsheet was produced on HP's --

MS. MANESS: That's right, Your Honor. So we had a detailed spreadsheet that showed all of the demos that HP has done.

What we don't have documentation of is what files were printed at each of those demos. HP does have a number of stock jobs that it keeps. One of them -- one out of the twenty or so is a document that contains variable data and is stored in a PDF file. We have produced that to IPT. And so we have provided the documents that we have that identify instances in which HP has used the press. And to the extent we could, we provided that one variable data job.

THE COURT: All right. Okay. Ms. Richards, what do you want on this issue of HP's use?

MS. RICHARDS: What would I like on the issue of HP's own use?

THE COURT: Yes.

MS. RICHARDS: The documents identified on HP's witness list manually. So looking at the spreadsheet that

Ms. Maness just talked about, the spreadsheet no doubt exists, but it doesn't file types, it doesn't show whether they're variable data jobs. Ms. Matioli, and I believe one of Ms. Maness' partners, said that the way to figure out if the jobs are one of the infringing file types of variable data is to look at the print files themselves. So we'd like those print themselves.

Ms. Matioli also testified that there were other print jobs that HP printed at their customer center, their demonstration center and their training center. So we would like the files themselves or the information about whether they were infringing files and the documents that show what customers they were demonstrated to. If they're using this infringing feature to sell the printers, we're entitled to know about that. So there's a bulletpoint, itemized list of documents of what their own witnesses said existed.

THE COURT: All right. Is this only with respect to the Optimized PDF feature?

MS. RICHARDS: The Optimized PDF feature and PPML for VDP and JLYT for VDP. I'm comfortable limiting it to those three --

THE COURT: Okay. So what do you have that would allow for the plaintiffs to know that, Ms. Maness?

MS. MANESS: Your Honor, we have the -- the actual PDF file that -- the stock job. It's a PDF job that contains variable data. We've provided that to the plaintiff. And we have a number of other stock jobs that do not contain variable data which we can provide if needed. And then I think Ms. Richards has pointed to occasionally -- sometimes HP will print a customer-provided file. HP generally does not keep those as part of the demonstration process. A customer brings it in. HP prints it. The customer can see the color quality, can see the substrate, the paper that's used. Other than that, we do not have any other documents. We've provided what we have on demonstrations we give.

THE COURT: Well, the dispute between you-all in this is whether, in fact, all of the data has been produced. I'm not still clear, Ms. Richards, on what it is that you think came up in Ms. Matioli's deposition that hasn't been produced or offered to you just now.

What is it?

MS. RICHARDS: Thank you, Your Honor.

So there are records of customer requested print jobs that HP has printed at its Graphic Arts Experience Center. They were obligated to produce -- obligated to preserve either the jobs or the records of the jobs starting two years ago. So they must have at least two years worth of those requests and documents related to the demonstrations.

They also have the Graphic Solutions Center, which is a training facility. Their witness confirmed there are documents printed there. The records of which ones were infringing and which customers were trained is what we're looking for. And, again, HP was obligated to begin retaining those. And if they're bringing in customers and saying, "I'll train you on how to use this infringing capability," that's --

THE COURT: Well, I doubt they're putting it that way. I doubt they're calling the customers in and saying, "We're going to train you on the infringing capabilities."

MS. RICHARDS: Right. So when they show them this is how you print a PPML file that include variable data, that's highly relevant.

THE COURT: Yeah, I hear you. I'm just -- I'm not hearing what documents you claim exist that are not produced. If you -- if your position is that there has been spoliation of some kind in connection with documents that did exist but don't exist any longer, I'll deal with that. But I'm not hearing an identified set of documents that exist that haven't been produced.

MS. RICHARDS: The print jobs themselves.

THE COURT: Okay. I thought you said you produced those already.

MS. MANESS: Yes, Your Honor.

MS. RICHARDS: They've produced one print job, one, after the motion.

The spreadsheet at A98 identifies -- I can't remember now if it's several hundred or several thousand different print jobs. There's no way to tell by looking at it which ones were infringing and which ones weren't.

THE COURT: I'm going to take a break until 20 minutes of, and I'll pick up.

(Recess.)

THE COURT: Okay. Let's review if there's anything additional that we have on IPT's motion to compel.

On the last question, this is what I'm inclined to do, and I'll let you-all comment on this.

I've got this -- this was sort of the tone through the discovery, there's a bit of plaintiff, or the defendant, depending on who is the movant, says, "I didn't get X" and the other side says, "Yes, you did," and there's this back and forth, "No, I didn't"; "Yes, you did." I'm not really in a position to resolve the "no, I didn't; yes, you did" kind of approach to things.

So on this question of HP's use, I think that it is appropriate for HP to produce the material that has been generically described here. I understand from you, Ms. Maness, that your position is we already did. And I think Ms. Richards is saying you didn't. And so I'm going to enter an order that says this is to be produced, and if you've already produced it, identify what you've produced. And that's what I'm going to do.

I'm not going to require that HP generate material that no longer exists. I heard the suggestion, Ms. Richards, that you think there has been documentation that did exist and no longer exists since the suit was filed. I'm not crediting that. That's your suspicion. And you may be right; you may be wrong. I don't know. But Ms. Maness, in the process of sort of recataloging what you've produced here, I do want you to make inquiry of whether you believe there are additional documents that are called for by this that no longer exist.

MS. MANESS: And, Your Honor, just so that I'm clear, are we being asked to produce any PDF files that may have been printed at the demo center that include variable data so that IPT can go and inspect them to see whether this obsolete feature is being used?

THE COURT: Well, I don't really understand, Ms. Richards, why you want that. It would seem to me that that would be extremely time-consuming. And the prognosis for your getting anything meaningful out of that is exceedingly remote. But I -- I think that that is a way for you to verify what is being said to you, if you want to spend your time doing that.

Do you?

MS. RICHARDS: I don't have a choice, Your Honor. What my clients are faced with are attorneys who keep calling the feature obsolete or --

THE COURT: So -- okay. So yes.

MS. MANESS: Your Honor, just so the record's clear, the files themselves are not going to indicate whether the feature was turned on, on the press.

THE COURT: Well, what are they going to indicate?

MS. MANESS: They're going to indicate whether they're a PDF or a PPML or a JLYT file.

THE COURT: Okay. So how do we establish in a way beyond the lawyers told me, that the customers are not, in fact, using the capitalized, Optimized PDF feature?

MS. MANESS: Your Honor, it's the testimony of our customer -- in the customer defendants' case, no, we have never turned on this feature. It's in the interrogatory responses which have all been given, no, we do not turn on this feature.

THE COURT: Okay. Let me back up for a moment, because I -- I think we shifted topics here, and I'm not sure if we did that intentionally or inadvertently.

So what the customers themselves are doing is one subject; what HP is doing is another subject. And I thought we had moved to the latter, what HP is doing.

MS. MANESS: We can talk about that. HP will print files on behalf of customers.

THE COURT: Right.

MS. MANESS: IPT has had the opportunity to take the deposition of the Demonstration Center's manager. And she confirmed in that -- she said "I don't know what Optimized PDF is." And if you look at the other testimony in the case, it makes sense why she didn't know, because nobody uses this feature. You have testimony from Alon Gazit over in Israel saying, "I don't know of anybody who is using it." You have Scott Clouthier, who is one of the main developers --

THE COURT: Okay. The answer may be nothing.

Do you have documents that reveal whether the feature is being used or not?

MS. MANESS: We do not. For any particular job, we do not have a record.

THE COURT: So you're saying the Optimized PDF feature could be used for a particular print job, but looking at the print job will not tell you whether it was used?

MS. MANESS: Correct.

THE COURT: Okay.

MS. MANESS: This is -- to be clear, this is a box that you check on the user interface. It's one of six different tabs with different configurations, "I want the color settings to be this rich," and then there's a box on page composition, the page composition tab that says "enable Optimized PDF." And it comes unchecked. And the question is, is it ever checked.

THE COURT: Okay. Well, how -- how are the print jobs -- if I order the production of the print jobs to you, Ms. Richards, how is that going to prove to you one way or the other whether the Optimized PDF feature was used by HP for its own printing, either for itself or for its customers?

MS. RICHARDS: The production of the PDF print jobs would be a very important clue, because the of the PDF itself and the variable data. We would be able to look at the PDF to see if it references this thing called an X object. You would see there's not just a PDF but this file of addresses. That would be a very helpful clue on the PDF side.

On the PPML side, you know that if the file itself is PPML and that there's variable data referenced or attached, other different ways to do it, then you know that that's one of the ones we're looking for. So we think that the print job itself should --

THE COURT: Okay. Is there a problem with producing them?

I'm highly skeptical, but I can't say from a technical perspective -- I'm saying that from my lofty perch up here -- that I think it is exceedingly unlikely that this is going to be terribly productive. But I haven't heard much about the burdensome nature of this production. So I'm inclined to allow it, because if Ms. Richards thinks she can build the case she needs out of this, then I'm inclined to let her do it.

What difficulty does this entail?

MS. MANESS: Your Honor, we can do it for HP. I'm concerned about the proportionality here. It's "give us everything you have to prove a negative." Even if there are X objects referenced, our systems are agnostic to the use of X objects. And there's testimony after -- days and days of testimony from Global Graphics and HP on this. X objects don't matter.

Settings that aside, if HP produces the 20 jobs it has or 30 jobs it has, that's one thing. If you are asking a company like O'Neil to produce every PDF it has processed that contains variable data printing or variable data content --

THE COURT: At the moment, we're only talking about HP producing --

MS. MANESS: My concern --

THE COURT: -- files related to what HP did itself.

MS. MANESS: And my concern, Your Honor, is it sets the standard for then saying, "Okay, now we want every single PDF from O'Neil."

THE COURT: Well, that's what I'm here for.

MS. MANESS: Please listen to my concern, is what I'm saying.

THE COURT: I have a really good memory. So you can put that in your -- in your quill, if that happens, that this doesn't mean that I'm going to go on a data review rampage.

MS. MANESS: Thank you, Your Honor.

And if I may, in the case of O'Neil in particular -- and I just want to showcase this for you, because it is very much a concern on behalf of defendants. If you're talking about every PDF job to prove a negative, whether it's productive or not -- again, let's set that aside. I also believe that it's not productive at all. You're also dealing with files that are health care files, credit card statements, and now are we talking about producing every PDF file? Because that's all O'Neil does.

THE COURT: Well, the more material that's produced that is consistent with the absence of the Optimized PDF feature, the less likely it becomes that I'm going to require more production.

And I'm going to come back to this the way I put this earlier when I was talking to you, Ms. Richards. This Optimized PDF, with all caps, and the Optimized PDF without caps, I'm not sure I know what else there is that is like Optimized PDF that's not Optimized PDF.

And as I said, I don't know what Judge Stickney was thinking. And when I put it that way, I'm not saying it in a negative. "I don't know what Judge Stickney was thinking," said negatively, is not what I am saying. I just don't know what the argument was, what counsel said what his premise was.

I'll give him an opportunity, if he wishes, to clarify that.

So all I'm ruling right now is that HP needs to produce files of the type that we have described. Not ruling out O'Neil, not ruling on protective order issues about O'Neil. From what I am hearing today, my strong sense is that this Optimized PDF issue is likely to go away in the case. And I'm going to give Ms. Richards a reasonable opportunity in connection with the HP material to see what's out there, if anything. And anything else beyond that has to be run through me.

MS. MANESS: And, Your Honor, for clarification on the files that -- again, I have to confirm whether HP has them or not. If HP has customer files that contain sensitive information, are you asking HP to go get clearance from those customers if there's a confidentiality obligation in place?

THE COURT: So are you -- we didn't make that clear, and I didn't get it.

Ms. Maness, are you saying that HP might have printed for a customer -- not O'Neil, but another potential -- a customer, one of the 300, Jeppersen --

MS. MANESS: Yes.

THE COURT: -- a set of patient records from Medical City Dallas?

MS. MANESS: It is possible. My concern is that some of this material may be sensitive to the customers.

THE COURT: Okay. What -- at this point we're speculating about whether that exists. If that exists, I will permit HP to redact sensitive personal information, subject to advising the plaintiff what sensitive personal information has been redacted. If there's -- and you'll retain the unredacted. And if there's a dispute about it, somebody will come see me.

MS. MANESS: And to be clear -- and I'm sorry to stay on this issue. When we do those redactions that -- we have to alter the file, and so whether there's X objects in it, you may not be able to tell that anymore. I just want to make clear to the Court what the -- what may result.

THE COURT: Well, I -- I'm going to have to -- because I think your, to an informed degree, speculating a bit about that. If that happens -- if you have a technical person who says if we pull out the name John Smith, then this feature that Ms. Richards is interested in looking at will go away, then we're going to have to reconvene by phone. And what I would likely do in that circumstance is allow it to be produced as is, with a severe limit on who can look at it and require that only one copy be retained, et cetera, et cetera.

But if I'm ordering it produced and you can't produce the information that might be productive because of the confidentiality, I'm just going to limit the field of who looks at it, not not require it to be produced.

MS. MANESS: Thank you, Your Honor. I think I understand your ruling.

And on the point about O'Neil, I do think we will need to bring that up with Judge Stickney, because that is a concern. And something that is required in his order, essentially, is produce all of the PDF files.

THE COURT: Okay. Well, I think this issue of Optimized PDF is very much in play now in light of what's been said on the record.

And so this is transparent, we're going to be in touch with Judge Stickney, not to tell Judge Stickney how I want him to come out, but to tell him what has happened today, to tell him he's going to get a copy of the transcript, and to tell him that he is likely to get a motion and what it's about. I'm going to tell him that or Becca will communicate that on my behalf to him. So he's not going to first know that something is coming when he gets it from you.

MS. MANESS: Thank you, Your Honor.

THE COURT: But I'm not going to argue the case of how it should come out. I'm not capable and nor would I.

MS. MANESS: Thank you, Your Honor.

THE COURT: All right. Ms. Richards, are we done on your motion?

MS. RICHARDS: I just want to make sure that I understand the ruling with respect to RP 8 that the request wasn't limited to just the print jobs themselves, although that's very helpful; the request -- the documents identifying instances in which HP practices the method, trade shows, tech centers, sales centers, and the circumstances surrounding the instances. We've asked for the customer presentations which would indicate which customers the features were demonstrated to.

HP's witness Bob Raus said they feature like Optimized PDF in all of their sales materials. We're asking for those sales presentations and documents and the training materials and documents.

THE COURT: Okay.

MS. MANESS: Your Honor, my --

THE COURT: Just a minute.

My understanding, Ms. Maness -- correct me if I'm wrong. My understanding from what you said is that one could find a sales document, which I assume has already been produced, marketing kind of material --

MS. MANESS: Yes.

THE COURT: -- that says we've got this Optimized PDF feature that will do X. My second understanding is that nobody had a demonstration on the Optimized PDF, and your position is that nobody used the Optimized PDF. And HP did not train or sell -- I'm going to put a little footnote on train -- on Optimized PDF.

MS. MANESS: That's correct, Your Honor. And to the extent that Ms. Richards said that Mr. Raus testified that Optimized PDF is feature in the marketing material, that's absolutely not true. In fact, Mr. Raus testified that "The Optimized PDF feature is obsolete. I know of no one using it."

THE COURT: So I'm just going to leave it at this.

If the Optimized PDF feature is in marketing, sales or training materials, those materials that describe that will be produced. If the HP products have been sold or marketed based on the Optimized PDF feature, those documents showing that will be produced.

I hear you telling me that it didn't happen; that it's a feature that there's there that nobody uses and nobody is selling on it. I don't know if it's true or not, but if it is in the source of training, sales or marketing, then the material will have to be produced.

MS. MANESS: Understood, Your Honor. And I believe we have produced what limited we do have on the feature, but we will go back and confirm based on your order.

THE COURT: So I think that's it on the motion of IPT to compel.

On your motion -- are you arguing everything, Ms. Maness?

MS. MANESS: Yes, Your Honor.

THE COURT: Just sitting there mostly quiet, you two.

MR. REINES: Sounds like it's better if I stay out of it.

THE COURT: The record will reflect I refused your inducement.

Okay. The Court, as I've stated, I am going to require that the plaintiff modify its infringement contentions to reflect what is actually claimed to infringe. And that's been stated on the record today, and the infringement contentions should be amended to reflect that.

I'm not going to require at this juncture, Ms. Richards, that you make the call about the Optimized PDF. I've stated that it's my educated guess that that's going to fall out. But you get to look at the materials I've ordered to be produced before you decide that. But the rest of it, which product is being challenged based on which series of patent families, that amendment is required promptly. I will say a week from Friday. So I want those infringement contentions modified to reflect that.

Now --

MS. RICHARDS: Your Honor, may I ask a question about your ruling? Because it's our position that the infringement contentions do fully reflect what's accused of infringement. They don't use the word "Optimized PDF," but as discussed today, the case is not limited to that.

They make very clear what about the PDF -- what about the use of a PDF is -- I'm happy to walk through the contentions.

THE COURT: It's not necessary, but I'm thinking you didn't hear me.

MS. RICHARDS: I don't understand your ruling, Your Honor.

THE COURT: Okay. The Court took your comments today about the variable data printing and inkjet patent families and Indigo for variable data printing and PWP for inkjet to be your statement as to what was claimed to infringe.

MS. RICHARDS: Yes, Your Honor. I understand what you're saying.

THE COURT: Well, that is what I'm saying.

I said just now that the issue of Optimized PDF and how that fits in your claims, I am not requiring you to do anything with your infringement contentions now, because productions have been ordered that will or will not confirm whether you have such a claim.

When those productions occur and you make a judgment about that -- and I'm not going to provide a date at this moment -- you will, if you limited your claims, further modify your infringement contentions, okay? Is that clear?

MS. RICHARDS: Yes, Your Honor.

Are you requiring that we -- they've requested that we summarize and include all of the evidence from the case.

THE COURT: Okay. I'm -- since you asked me a question about what I said, which was the 30,000-feet point, I'm now going below 30,000 feet. So wait.

MS. RICHARDS: Oh, thank you, Your Honor.

THE COURT: Ms. Maness, on the issue of the Optimized PDF and how that shakes out in the lawsuit, you may come back to the Court at some reasonable time after these productions are done and say -- okay, you first confer with Ms. Richards and she says, no, still think it's in the case, and you say it's not in the case, "Judge Lynn, here are 50 reasons why; 50 things we've produced; 100 things we've said under oath. That's out. Make them amend to drop it." So I'm putting that issue to the side, because it's premature in light of fact that there's additional discovery being ordered.

MS. MANESS: Your Honor, could we ask for a clarification? Because there's really two issues with Optimized PDF and the contentions. One is the contentions as worded right now accuse all PDF, Optimized, capitalized, optimized, small, any PDF of infringing.

And from what we've heard from the plaintiff today and their conduct in discovery has showed is the focus really is, is this feature turned on or off. And so the contentions should say -- and I'll get to the fact -- factual basis in a moment. But the contentions should say if the feature is turned on, X, Y and Z. Step C is performed because X, Y and Z; see source code.

Right now the contentions don't even do that. And I understand --

THE COURT: Well, I think we're arguing in a circle here. Because I made this point that I wasn't clear whether the discussion before Judge Stickney was limited to the capitalized Optimized PDF or the lower case something else I don't know. So I can't very well require that the infringement contentions be amended to reflect that issue until that issue is sorted out.

MS. MANESS: My concern is that the -- the plaintiff has taken the position -- and I may be corrected here -- but that the allegedly infringing use occurs when the Optimized PDF feature is turned on. And there is no infringement, alleged or otherwise, when it's turned off.

THE COURT: Okay. So I'm just going to say this about that, Ms. Richards.

When you amend your infringement contentions by a week from Friday, you will state what it is that you claim is infringing about PDFs, whether it has only to do with the turning on the feature or whether it has to do with something else, and if so, what.

Okay. All right.

MS. MANESS: Thank you, Your Honor.

So, Your Honor, if we can get down more into the details, we all know that the local rules here require infringement contentions. And the purpose of those contentions is to provide notice of the specific theories of infringement and to streamline discovery. Those are two key, important points.

IPT served its amended contentions back in March, about three months ago now. I think Your Honor has already address the Point 1 here on the screen, the contentions still allege infringement by the PWP presses against variable data -- the variable data patents. That's going to come out.

From what I understand, Your Honor, the second question is do the plaintiffs need to focus on the actual file types used by the customers and is that one of the issues that we're reserving or is that -- will that be required by your order. For example, O'Neil uses only PDF. And right now the contentions say that O'Neil could use any of a number of file types.

THE COURT: The contentions say that O'Neil does use different file types?

MS. MANESS: Yes.

THE COURT: Well, it's just that I don't think that the interrogatories are the place to sort out that kind of

CONFIDENTIAL - ATTORNEY'S EYES ONLY

difference.

What I want the interrogatory to reflect is what is claimed to infringe; what types of files are claimed to infringe what.

And if the result of that is that O'Neil has conclusive evidence that it doesn't infringe, because the files that have been identified are not consistent with O'Neil's approach, then that is the subject of a motion. It's not something that I think needs to be set out in the interrogatory. Where they say O'Neil uses this kind of file and your position is no, they don't, that's just not the kind of dispute that I think should be resolved through interrogatory answers.

MS. MANESS: So the expectation, Your Honor, in the contentions is if O'Neil uses PPML, it would infringe because and then to lay it out.

THE COURT: Yes.

MS. MANESS: I understand.

Then I will spare you the interrogatory response and the deposition testimony that shows that O'Neil uses only PDF.

I think we've discussed the Optimized PDF, and Your Honor has directed the plaintiff to identify how exactly it is they're accusing PDF infringes.

THE COURT: I'm not -- to shorthand this -- this is in effect what you were asking me, Ms. Maness. I do not agree with the approach in your motion that the interrogatory is the place where I require a resolution of the facts.

To the extent that you're asking what their position is on infringement, your example of O'Neil is a good example. If O'Neil does acts, then O'Neil infringes. If you can prove that O'Neil doesn't do acts, then additional action is appropriate. But I don't think that this is the place to sort that out.

MS. MANESS: Your Honor, if I may point out -- and I understand that you're not addressing Judge Stickney's ruling. But if I may point out the disparity here is IPT right now is allowed to allege use of multiple different file types. And even though O'Neil uses only PDF, O'Neil has to respond in the non-infringement contention interrogatory that Judge Stickney ordered be responded to in full, why it wouldn't infringe with PPML? Or does O'Neil just need to say "We don't use PPML"?

We're concerned about the burden here where plaintiff is allowed to keep its contentions very broad, and then it puts this burden on the defendants of, "Well, if we used PPML, we wouldn't infringe because of X, Y and Z."

THE COURT: Well, I don't know how to resolve that any further than I have, Ms. Maness.

I'm going to say it one more time, and then I'm done saying. I'm not redoing Judge Stickney's order. So if the contention were -- that has been made, that the discovery that has been ordered is burdensome because it's premised on something that is undeniably inaccurate, then that was an issue to be resolved on the motion before Judge Stickney. I'm not redoing that.

MS. MANESS: Understood, Your Honor.

If I may address another issue that we've had with the contentions, and it's the issue of the absence of source code. This is very much a software-heavy case. All of the activities being performed within the computer on the HP press are software activities, and source code is necessarily involved.

And we're at a point in the case where the plaintiff has had plentiful opportunity to look at the source code. It's been available for 14, 15 months now. They have spent three days with it. It's time for them to point out in the source code where they believe that each of these steps is performed. And case after case make that clear.

THE COURT: Okay. I'll hear a response on that.

Anything else on your motion?

MS. MANESS: Not at the moment, Your Honor.

THE COURT: Okay. Ms. Richards, why shouldn't I have you do that?

MS. RICHARDS: Are we talking about the source code aspect, Your Honor?

THE COURT: Yes.

MS. RICHARDS: My client, IPT, is entitled to prove infringement using any evidence. It's not required to use source code. It can use deposition testimony, it can use documents, it can use third-party documents. It can use any evidence.

The cases that the defendants -- and so the idea of forcing us to prove the case through source code -- source code is the most expensive. The code is in California. It's under strict protective order requirements. It requires us to have an expert who is quite expensive to review it. It's not something a lawyer can do. And it's not something you can even print out. There's absolutely no reason to require us to prove our case using a certain type of evidence when we can prove our case using any number of kinds of evidence.

With respect to their cases --

THE COURT: Okay. Let me ask you a question about that.

MS. RICHARDS: Uh-huh.

THE COURT: Is that to say that you're now conceding that you are not going to attempt to prove your case through source code?

MS. RICHARDS: No. In two weeks, I have responses from HP in response to 35 requests for production of documents coming. We've been trying to get those documents for two years. I don't know what's in there. If everything that I need is in those documents and the repeated 30(b)(6) of their witness that I'm entitled to take and they're actually required to pay the costs for, if everything I need to prove on claims is in those documents or in that deposition, then I'm not required to review the source code. If it's not in there, I'm entitled to review the source code.

Our situation is different from all the cases they've cited. In these cases people said "I can't give you my infringement contentions until I review the source code." We have never, ever said that. We've said, "These are our contentions." If you review the contentions, they're very specific. We've never said we're holding back until we review the source code.

THE COURT: Well, let me make this clear. If you're going to claim that there is infringement because of the contents of source code, then you're going to have to walk through how the source code infringes. If you don't, you're not going to be able to prove your case up.

MS. RICHARDS: I agree with you, Your Honor. But I don't agree that the right time is now when they're in the middle of discovery.

THE COURT: Well, I wouldn't call it the middle. I would call it the looming end.

MS. RICHARDS: Well, that's true from one perspective, Your Honor. But the truth of the matter is, we've been trying to get the same documents and informed deposition testimony out of these defendants for two years. And we've sent them 50 letters. And Judge Stickney's order reflects HP alone is required to respond to 35 requests in two weeks, answer five interrogatories.

And so from our perspective, Your Honor, I'm sorry to say it, but in some sense we are at the beginning of discovery. I don't know what's going to be in their documents; I don't know what 30(b)(6) is going to say once we --

THE COURT: Okay. Here's where I'm leaving this for now. I am not requiring that plaintiff identify with citations to source code how the accused products infringe and methods infringe.

However, I will, upon further motion, require that. And if that is not done, the source code may not be used as a method of proving infringement. So I'm going to defer that until some reasonable time after the conclusion of discovery.

I don't agree with, I don't accept the premise that we're at the beginning. We are really close to the end, and that will be reflected in my rulings to go.

Yes?

MR. JOHNSON: Your Honor, with respect to your ruling, I would only -- I want to make this one point, and perhaps it will not change your mind. But in this case, everything that is -- that would infringe has to occur in the source code.

So it's not as much a matter of do they need it to prove it. It is essential, because it actually says what we do. And the best example I could come up with is, Your Honor, no matter what the inventor says he invented -- he can say, "I invented the football, I invented the football," and he can write documents, "I invented the football." But if the claims show that he invented a basketball, no matter what he said, those claims control.

Our documents, our witnesses can say all it wants to about how we perform certain functions. But what actually shows how we perform it, and the only thing that really shows what happens inside that machine and whether it infringes or not, is source code.

And so it's not a matter of do they need it. It's a matter of that -- that's core to the case. And if they can't point out in the source code where there's infringement, then the case should go away.

THE COURT: Okay. I hear you, Mr. Johnson. I'm receptive to the issue that you've raised but not now.

MR. JOHNSON: Thank you, Your Honor.

MS. MANESS: Your Honor, may I make one more final

CONFIDENTIAL - ATTORNEY'S EYES ONLY

point about contentions?

THE COURT: Yes.

MS. MANESS: Since they --

THE COURT: About?

MS. MANESS: About the contentions.

THE COURT: Yes.

MS. MANESS: Since they are being revisited, on the sync contentions -- and I know we've spent most of our time on the variable data aspect of the case -- on the sync contentions, we are concerned, and part of our motion is, they have not met the required notice function under the local rules.

And I've pointed -- we've pointed the Court to two examples in our briefing, and they're up here on the screen of. One is the element "a plurality of print engines." And IPT points to three different components, all of which are a plurality of something. It doesn't say which ones it contends meet that claim limitation, which leaves us guessing, well, is it print bars, is it print bars minus something? What is being accused? And that's the whole point of contentions.

THE COURT: Okay. I agree with --

MS. MANESS: Thank you, Your Honor.

THE COURT: All right. That modification needs to be made, Ms. Richards. That's not clear what you're contending to be a print engine. If you are claiming all of those are print engines, then say that.

MS. RICHARDS: Okay. All right.

THE COURT: Okay.

MS. RICHARDS: Okay. That's -- in our briefing we said we're entitled to contend that multiple components constitute the print engine and we've identified them --

THE COURT: Yes. If that's what you're saying, then say it.

MS. RICHARDS: Okay, Your Honor. I understand your ruling.

MS. MANESS: Thank you, Your Honor. That problem exists for several of the other elements in --

THE COURT: Well, I'll just say generally, then, without spending the time to go through these, that, Ms. Richards, you're going to be amending these anyway. If your position is that different words all mean what is in the claim, that any and all of them mean, in this example, print engines, if that's your position, then say that. I can't -- I can't read this on its face and draw the conclusion that you mean that multiple -- that print bars, printheads and the inkjet dyes all individually and collectively mean print engines.

And so it's, in effect, a summary of what your position apparently implicitly is, but it doesn't say it explicitly, and I want you to say it explicitly.

Okay. What else have we got?

MS. MANESS: Your Honor, that's all from us, although Mr. Reines has -- briefly.

MR. REINES: Your Honor, just two points I'd like to make, and I do it reluctantly, but I know that my client would want to me to do it. There's two elements of the way things have played out that feel unfair and maybe just catharsis. And I'll be succinct with Your Honor.

In the motion that was brought against us that was decided by Judge Stickney, we were ordered to update our non-infringement contention interrogatory with current discovery, okay?

We are asking to compel the plaintiffs to update their infringement contentions with current discovery. It can't be appropriate for the non-infringer to have to update based on current discovery and not the other way around.

THE COURT: Where does it say that?

MS. MANESS: It's interrogatories -- and we can break them out for you. I think it's 2, 4, HP.

THE COURT: I'm sorry. I'm looking at Judge Stickney's order. Tell me where it says what you just said.

MS. MANESS: At the end, Your Honor, it calls for HP to answer Interrogatories 1, 2 and 19. IPT asked for clarification on Sunday evening, and Judge Stickney provided it on Monday, that Interrogatory 2 means both MDL

CONFIDENTIAL - ATTORNEY'S EYES ONLY

Interrogatory Number 2 and Number 2, which is the non-infringement interrogatory. And then it also requires that the printer defendants, on the last page, fully answer a list of interrogatories. Within that list are non-infringement interrogatories to each of the printer defendants. I believe it's Number 4.

MR. REINES: And this was much of the briefing, Your Honor. And we were asking that the issue before Your Honor be decided first, so we figured out what infringement contentions had to be made before non-infringement contentions.

THE COURT: Okay. I'm going to take that issue under advisement.

MR. REINES: Thank you. And one other issue, Your Honor. And again, I really appreciate how hard you've worked on what can be tedious issues. We all know that.

The second issue is -- and it's just back to Optimized PDF. Just give me two minutes, and then we'll scoot to the airport and so forth.

The request that we've been ordered to answer, for example, O'Neil, okay, is variable data printing without any limitation to anything. Just -- there's a laundry list of JLYT, all of the different things. All they do is PDF with no OPDF. That's 30(b)(6) testimony.

If we have to produce documents including -- it's not just the technical documents so they can determine if there's some secret lower cap -- you know, maybe they went in and changed the source code of their press, something like that, right? But it's marketing plans, business plans, profitability, all financial, marketing documents about all variable data printing. It's a variable data printing company. Everything they do is variable data printing.

They happen to make the PDFs -- you know, make them all -- a hundred of them or a thousand of them when they do that upstream.

THE COURT: Have a seat, Ms. Richards.

MS. RICHARDS: Thank you, Your Honor.

MR. REINES: So they -- what their value add is, if someone has a variable data document that goes a hundred different companies, is they do all of that magic up front and then put in the PDFs and give a hundred-page PDF that's treated just like static data and send it all to the press. And the press just does it and just rips one after the other.

In the status report, we put a declaration in. They've taken testimony from London to here, everywhere. There really is no doubt that unless Optimized PDF, lower case or upper case, is engaged, there's no reuse.

For us to have to produce basically the entire business development -- business plans for this customer of ours, because they won't limit their claim to actual Optimized

PDF, which would actually be what they care about -- I don't know why they want to know about all variable data printing in this huge company when it's all PDF printing. So we have that -- large, and it's going to require settlement that -- yes, we'll file a summary judgment and, yes, oh, PDF is going to drop out and, yes, it's a nothing issue, but it's going to soak us to incredible lengths. 35 -- I think she was bragging about 35 different categories of documents. If it's confined to OPDF, then we're actually discovering what people want. I just don't understand why we would have to do all PDFs. It's the whole company.

THE COURT: Okay.

MR. REINES: Thank you.

THE COURT: All right. Ms. Richards, I'm going to hear from you on that issue.

Do you actually want all PDFs?

MS. RICHARDS: No, Your Honor. I don't believe that that's what we asked for or Judge Stickney's order --

THE COURT: Well, what do you want? In that regard, you want what?

MS. RICHARDS: I don't even know what we're talking about, Your Honor. What document request are we talking about?

MR. REINES: 1 through 35.

MS. RICHARDS: Each request has a specific thing that it requests and none of them request the production all files or of the entire business. I don't think that we should reargue the motion that Judge Stickney has --

THE COURT: Well, I'm not attempting at this moment to reargue it. I'm trying to narrow the focus of the dispute to save time for Judge Stickney and ultimately for me, for reasons that are not at all clear.

I'm not sure if you-all are actually fighting or not fighting about what magnitude of and types of files are to be produced.

So this is the way we're going to sort this out.

Ms. Maness, you're making me seasick. Stop.

MS. MANESS: Sorry, Your Honor.

THE COURT: What I want to have you do -- and I'm going to start with the defendant. This is not to reargue before Judge Stickney. I want a maximum three-page, double-spaced document from the defendants explaining how you think 1 to 35 is requiring you to produce material related to all PDFs, okay? And I want you to do that -- can you do that by Thursday?

MR. REINES: Yes, Your Honor.

THE COURT: Okay. By the end of the day on Thursday.

By the end of the day on Tuesday of next week, Ms. Richards, I want you to explain, "Yes, indeed, I want all of that," or "No, I don't really want that, and here's what I really do want." Because I'm not sure you-all have a dispute about it.

The Court will, only to this extent, beyond -- only to the extent that the enumerated interrogatories -- I'm sorry -- Requests for Production 1 to 35. Only to the extent that there is a dispute about this issue, the Court will defer the production very slightly.

My understanding -- this is only Bulletpoint 1 on Page 3 of Judge Stickney's order. It's the first part of Bullet 1. As to those only, the Court will -- the Court will add one week to the production. And that will allow this issue to be teed up. And I'm doing that not because I disagree with anything that Judge Stickney decided, but I don't know if there's a genuine dispute about this, and I want to know and I want Judge Stickney to know.

So the only extent to which I am modifying Judge Stickney's order is to add one week to complete the production of documents in response to 1 to 35.

MR. REINES: One minor thing?

THE COURT: No, not until I'm done talking.

MR. REINES: Okay. Sure.

THE COURT: And you will -- you should -- in an abundance of caution, you should probably start compiling that, even though you think it's burdensome, because I'm not suggesting to you that at the end of the day, you won't have to.

Now, what did you --

MR. REINES: Thank you, Your Honor. Apologize for the interruption.

I think the same is true for HP. In other words, if it's all PDFs versus not all PDFs. So it's the first bullet on Page 2 under the HP. It's just the same issue.

THE COURT: I don't know what you're referring to.

MR. REINES: If you turn to Page 2 of the order, there's the three bullets under HP. And the first bulletpoint is parallel to the bulletpoint on -- with respect to the customers. It's just the customer and then HP.

THE COURT: Well, if you're representing to me that this issue is the same in Bulletpoint 1 on Page 2 and the first part of bulletpoint -- the first bulletpoint on Page 3, the Court will rule the same way on both. That is, I'm extending the production by one week of those materials, and I'm ordering the exchange of what is required and what should be required. No replies. One shot from each of you, to be finished by next Tuesday.

MR. REINES: Thank you.

THE COURT: Okay. All right. Anything else?

MS. MANESS: Your Honor, very briefly, just on logistics.

THE COURT: I'm going to hear from Ms. Richards.

MS. RICHARDS: May I be heard for just a moment on the unfairness of the mutual exchange of contentions and current discovery that Mr. Reines brought up?

THE COURT: Well, yes, I mean, you can, but it's an academic question. But go ahead. I'm not resolving that question right now. I said I'd take that under advisement. If you want to be heard on it, since I'm considering it, go ahead. Sure.

MS. RICHARDS: Your Honor, the case law that we've provided in our briefing makes clear, infringement contentions, be they preliminary or final, are not the appropriate place for any including any evidence. The purpose is for the plaintiff to disclose the notice of what they're claiming and their theories of infringement.

Defendants have always and at all times known how their own machines work and how their own defendants use the machines. They know what evidence they've provided to us. They know what their deponents have said. They know what their documents say.

This idea of having us include their evidence in our contentions is not what the contentions rules require. And additionally, it would require nearly constant updating from my client.

I've asked them a number of times, when would this -- at some point in discovery -- every time we find a document we have to add it in?

THE COURT: Okay. Let me interrupt you for a minute. I'll let you finish your point. But when I said I was taking this under advisement, I was taking it under advisement not because I'm inclined to make you produce more. I'm going to look at the question of whether it is fair not to require you to produce evidence according to the contentions but to require the defense, in their non-infringement positions, to produce evidence. That lack of parallelism has my attention. Not because I think you should produce more. The issue is whether they should have to produce something that you don't have to on the infringement side. So if you wants to be heard on that, go ahead.

MS. RICHARDS: Thank you, Your Honor. That's very helpful.

So the reason why we moved to compel on the non-infringement contentions is because we had an issue arise where we went to an HP deposition, and for the first time a witness told us a contention we've never heard before in the middle of a deposition.

The purpose of the non-infringement contention interrogatory is for defendants to tell us why they don't infringe and give us -- and identify the documents that they think supports their case so that when we get to the deposition, one, it's not prejudicial to us to hear something new and, two, because it's infinitely more efficient when we ask about what is actually at issue in the case. That's the point of asking them what the basis is for their contention and to have them identify --

THE COURT: Okay. I don't think we're -- we're not talking to each other. I get that. I agree that you should have non-infringement contentions; they should have infringement contentions.

The question that Mr. Reines raised -- and I'm sorry -- you know, I know you; it's Reines (pronouncing). I'm so sorry. It's your partner that messes me up, because I have to remember Reines and Maness (pronouncing). So I'm just going to talk to Mr. Johnson.

The issue is why should you not have to produce documents and evidence supporting infringement, but they have to produce documents and evidence supporting non-infringement. Why does that make sense?

MS. RICHARDS: I'm not producing any documents and evidence supporting infringement. I'm relying on their evidence. I've disclosed my contentions based on publicly available information. I'm then relying on evidence that they're producing to support my contentions, which will be in our experts reports.

There's no reason for me to give them ongoing updates about what documents they're giving me. It seems to me -- obviously, the Court can rule otherwise. But that's an entirely different exercise than them telling me I don't infringe, and the reason why I don't infringe is because this document here shows I don't reuse -- or whatever the document might be. It's a different exercise for them to come forward with their evidence, than for me to say, "Here's what the evidence you just gave me says."

THE COURT: Okay. I'm taking that under advisement.

MS. RICHARDS: May I bring a totally unrelated issue solely to the Court's attention, not obviously to take up any more of your time with argument, but for my client?

THE COURT: Okay.

MS. RICHARDS: It relates to -- you may be aware of it already. It relates to the protective order and the upcoming mediation. Are you aware of this issue, Your Honor?

THE COURT: I'm only aware of what is brought to my attention. Where would I learn this from?

MS. RICHARDS: We relatively recently filed a letter attempting to bring this to your attention, but I understand that the Court is busy.

The issue is, Your Honor --

THE COURT: Hold on. I'm busy, but I'm not in the habit of just letting things go past me. Are you saying you filed a motion in connection with this?

MS. RICHARDS: It's their motion that they filed when the case was in the Eastern District of Texas. And it's been sitting for 18 months. We've asked them to bring it to your attention. They've declined. But it puts us in a very awkward situation, because in a couple of weeks there's a mediation. And as long as their motion is pending, we cannot share anything about the case with our client, including the management.

So how many days ago do think we filed the letter, two or three or four?

After their last rejection of our request that this be resolved, we thought in anticipation of the hearing we should at least bring this to the Court's attention.

THE COURT: Okay. Well, I'm unprepared on this, so I'm not going to do this on the fly when we're five minutes past the time that I've set. So I will read your letter, and if I want a response, I will direct it.

MS. RICHARDS: Thank you, Your Honor.

MS. MANESS: And, Your Honor, I just want to make sure that the transcript is designated as confidential here. Those job origin reports are highly sensitive and --

THE COURT: Okay. Again, well, there are people here in the courtroom other than the people up here, who I know. I don't know who the two people are in the back.

CONFIDENTIAL - ATTORNEY'S EYES ONLY

Do you?

MS. MANESS: I don't.

THE COURT: Okay. Who are you two?

UNIDENTIFIED: We were invited to sit in. We spoke with Ronnie, I believe your court clerk.

THE COURT: He's not my court clerk. He's a court clerk, but he doesn't work for me.

UNIDENTIFIED: We're legal interns, legal students, and we were invited to sit in.

THE COURT: Okay. So this is on you-all. If something is going on in the courtroom that is confidential, you should be scanning the room. You-all bring people with you all the time, and I don't know who they are.

I'm not talking to you two when I say that.

Okay. Everything that happened in here today, if I asked you questions about it, don't be offended, but I don't think you could --

UNIDENTIFIED: No.

THE COURT: "So describe that VDP" and that --

Now, do you feel like if I asked you a bunch of questions about the secret nature of this variable data printing and sync, that you could articulate it?

UNIDENTIFIED: No, Your Honor.

THE COURT: Awesome. Okay. So you-all are welcome anytime, especially when you don't understand anything about this.

Okay. Yes, the Court designates it as confidential. And from now on, you-all are obligated to look around at the beginning, not at the end, to see if I need to give an instruction at the beginning.

MS. MANESS: Understood, Your Honor.

THE COURT: All of my interns are instructed to clear your mind if you had a sudden epiphany anytime between now and the start about what the heck we were talking about.

MS. MANESS: Thank you, Your Honor. Appreciate it.

THE COURT: All right. Thank you.

(Proceedings concluded.)

CERTIFICATE OF OFFICIAL REPORTER

I, D. Keith Johnson, RDR, CRR, Federal Official Realtime Court Reporter, in and for the United States District Court for the Northern District of Texas, do hereby certify that pursuant to Sections 753, Title 28, United States Code, that the foregoing is a true and correct transcript of the stenographically reported proceedings held in the above-entitled matter and that the transcript format is in conformance with the regulations of the Judicial Conference of the United States.

Dated this 9th day of June, 2016.

/s/ D. KEITH JOHNSON______________________
D. KEITH JOHNSON, RDR, CRR
TEXAS CSR NO. 3781
FEDERAL OFFICIAL COURT REPORTER
1100 COMMERCE STREET, ROOM 1572
DALLAS, TEXAS 75242
214.753.2325

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

INDUSTRIAL PRINT TECHNOLOGIES, LLC, PLAINTIFF V.	The Honorable Barbara M.G. Lynn
O'NEIL DATA SYSTEMS, INC. AND HEWLETT-PACKARD COMPANY	Civil Action No. 3:15-cv-01100-M
O'NEIL DATA SYSTEMS, INC. AND HEWLETT-PACKARD COMPANY	Civil Action No. 3:15-cv-01101-M
O'NEIL DATA SYSTEMS, INC. AND HEWLETT-PACKARD COMPANY DEFENDANTS.	Civil Action No. 3:15-cv-01104-M

IPT'S FIRST SET OF REQUESTS FOR PRODUCTION TO DEFENDANT O'NEIL DATA SYSTEMS, INC. ("O'NEIL") (NOS. 1-36)

Pursuant to Rule 34 of the Federal Rules of Civil Procedure, Plaintiff Industrial Print Technologies LLC ("IPT") hereby requests that Defendant O'Neil Data Systems, Inc. ("O'Neil") produce the documents and things requested herein for inspection and copying, to the extent not already produced, within 30 days of service hereof at the offices of Fitch, Even, Tabin & Flannery LLP, 120 South LaSalle Street, Suite 1600, Chicago, Illinois 60603, or at such other time and place agreed to by counsel.

DEFINITIONS

As used herein, the term(s):

1. "IPT" means Plaintiff Industrial Print Technologies LLC.

2. "HP" means Hewlett-Packard Company, all present and former officers, directors, employees, consultants, or other persons or entities acting in concert with them or acting on behalf of them or who are subject to the direction or control of the foregoing.

3. "Variable Data Print Job" means a printing process in which elements on a printed page may change between one instance of the page and another instance of the page in the job.

4. "Static Print Job" means a printing process in which the elements on a printed page do not change between one instance of the page and another instance of the page in the job.

5. "HP Variable Data Printing Presses" includes any printing press that is capable of processing Variable Data Print Jobs and that is manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP, including without limitation HP Indigo Digital Presses and HP Inkjet Web Presses enabled for Variable Data Print Jobs.

6. "HP Indigo Digital Presses" includes all models within the HP Indigo product line manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP (e.g., model W3250, 3550, WS4600, 5000, 5600, WS6600, WS6600p, W7250, 7500, 7600, 10000, 20000, and 30000 presses) and all components and subsystems thereof (e.g., paper supply, utility cabinet, printing engine, press computer, touch screen panel, stacker, etc.) and specifically includes each digital front end ("DFE") associated with each such press.

7. "DFE" means digital front end, and specifically includes the raster image processor(s) ("RIP") associated with each such press.

8. "HP Inkjet Web Presses" includes all models within the IP Inkjet Web Press product line manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP that contain multiple inkjet printheads (e.g., HP T200, T300, T350 and T400 inkjet web presses) and all components and subsystems thereof (e.g., print engine, ink delivery systems, service stations, in-line process monitor, press interface adaptor/frame broker, press controller, dryer, paper supply and rewind systems, etc.) and specifically includes each digital front end ("DFE") associated with each such press.

9. The "Accused Variable Data Printing Methods" means the methods performed to design and create Variable Data Print Jobs for HP Variable Data Printing Presses and to process for printing and print Variable Data Print Jobs using HP Variable Data Printing Presses.

10. The "Accused Inkjet Control Systems and Methods" means the components used in and the methods performed by the HP Inkjet Web Presses to synchronize multiple printheads of the press.

11. "Variable Data Printing Patents" means U.S. Patents Nos. 5,729,665 ("the '665 patent"); 5,937,153 ("the '153 patent"); 7,274,479 ("the '479 patent"); 7,333,233 ("the '233 patent"); and 6,381,028 ("the '028 patent").

12. "InkJet Patents" means U.S. Patents Nos. 6,145,946 ("the '946 patent") and 6,493,106 ("the '106 patent").

13. "Patents-in-Suit" means any of the Variable Data Printing Patents or the InkJet Patents, individually and collectively.

14. "Programming Information" includes programs expressed in a hardware definition language (e.g. VHDL or Verilog or others) as well as software programs written for storage or execution in a hardware component. "Programming Information" includes documents describing the development software and hardware system in which the program was created and documents identifying the manner in which the program is used within the component including the precise steps that must be taken to install the program in the component.

15. "Global Graphics" means Global Graphics Software Inc. and/or Global Graphics Software Ltd. and their executives, officers, and present or former employees, as well as related companies.

16. "Printer Defendants" means Cenveo, Fort Dearborn, O'Neil, Quad Graphics, and Vistaprint.

17. "Cenveo" means Defendant Cenveo, Inc. and its executives, officers, and present or former employees, as well as related companies.

18. "Fort Dearborn" means Defendant Fort Dearborn Company and its executives, officers, and present or former employees, as well as related companies.

19. "O'Neil" means Defendant O'Neil Data Systems, Inc. and its executives, officers, and present or former employees, as well as related companies, including O'Neil Digital Solutions, LLC and O'Neil Data Systems LLC.

20. "Quad Graphics" means Defendant Quad/Graphics, Inc. and its executives, officers, and present or former employees, as well as related companies.

21. "Vistaprint" means Defendant Vistaprint USA, Inc., and its executives, officers, and present or former employees, as well as related companies, specifically including, but not

limited to Vistaprint North American Services Corp., Cimpress U.S.A., Inc. and Cimpress Windsor Corporation.

INSTRUCTIONS

1. Continuing Obligation. With respect to each of the requests for production, unless otherwise expressly stated, the information sought is that which is current to the date of your answer. These requests for production are of a continuing nature. You are required to serve supplemental responses under the applicable rules if your knowledge changes in the future. Please take notice that objection will be interposed at trial to the introduction by Defendant of any evidence requested by these requests for production but not fully disclosed in your responses, including required supplementation.

2. If all or any part of a document is in a language other than English, identify and produce any translation of the non-English language document.

REQUESTS FOR DOCUMENTS AND THINGS

REQUEST NO. 1:

Documents sufficient to identify each specific HP Indigo Digital Press and/or HP Inkjet Web Press operated by O'Neil, and the physical arrangement and functional interrelations of all hardware and software components and subsystems of each HP Indigo Digital Presses and each HP Inkjet Web Press.

REQUEST NO. 2:

Documents sufficient to confirm O'Neil's operating procedures and customary practices for receiving and processing Variable Data Print Jobs and Static Data Print Jobs on its HP Indigo Digital Presses and HP Inkjet Web Presses, including documents establishing the complete work flow, from the creation and/or receipt by O'Neil of the page description specification for a print job through the fulfillment of the print job.

REQUEST NO. 3:

For each HP Variable Data Printing Press, documents sufficient to identify and describe the specific software applications used to practice the Accused Variable Data Printing Methods (e.g., Job Consumer, Indigo Press Controller, Global Graphics Harlequin), and how each software application is used.

REQUEST NO. 4:

For each software application capable of practicing the Accused Variable Data Printing Methods, documents (including software design documents, flow charts, product specifications, technical descriptions, protocol specifications, file specifications, format specifications, workflow diagrams, manuals, training information, etc.) sufficient to describe:

a. how the software identifies a static data area (i.e., an area for image elements that do not change from one instance of a page to another instance of the page) and a variable data area (i.e., an area for image elements that may change from one instance of a page to another instance of the page) of a print job described by a page description language (e.g., PDF, PPML, PPMLT, JLYT files, etc.);

b. how the software associates graphics state information (e.g., font characteristics, type size, text alignment, word wrapping, page registration parameters, fill color, angle, scale factor, etc.) with static and variable data areas;

c. how the software creates and stores bitmaps of static and variable data areas; and

d. how the software merges static data bitmaps and variable data bitmaps.

REQUEST NO. 5:

Documents identifying or discussing any standards for creation or processing of Variable Data Print Jobs that O'Neil and/or its customers use in connection with the Accused Variable Data Printing Methods, including for example, PDF/VT, PDF/X-4, PDF/X-5, PPML, PPMLT, or JLYT.

REQUEST NO. 6:

To the extent not produced in response to other Requests, documents, including, reports, manuals, drawings, brochures, texts, product specifications, design specifications and source code, sufficient to establish the design, structure, operation, features and/or functions relevant to O'Neil's use of the Accused Variable Data Printing Methods on the HP Variable Data Printing Presses.

REQUEST NO. 7:

Documents describing the services (e.g., technical support, training, installation, customization, integration, and/or maintenance) that any provider of software used on the DFEs of the HP Variable Data Printing Presses has provided to O'Neil relating to the Accused Variable Data Printing Methods.

REQUEST NO. 8:

Documents referring or relating to the advantages and/or value associated with the Accused Variable Data Printing Methods (e.g., improved printing speed, increased productivity, additional throughput, reduced cost, increased printing profits, etc.) relative to alternative approaches for running Variable Data Print Jobs.

REQUEST NO. 9:

Documents such as business plans, strategic plans, business strategy reports or marketing plans discussing the strategic importance, value added, target customer market, use cases, applications, and/or market potential of O'Neil providing Variable Data Print Job printing services to the marketplace.

REQUEST NO. 10:

Documents referring to, relating to or comprising estimates or forecasts of the demand for Variable Data Print Jobs printing services in the marketplace and/or O'Neil's competition for supplying such services, including any internal estimates, estimates in trade or other press, estimates by industry analysts or the like.

REQUEST NO. 11:

Documents relating to the advantages and/or value associated with the Accused Inkjet Control Systems and Methods of the HP Inkjet Web Presses (e.g., improved print quality and accuracy, avoidance of printing mismatches and other alignment errors on the printed page, improved printing speed, increased productivity, additional throughput, reduced cost, increased printing profits).

REQUEST NO.12:

All agreements between O’Neil and HP for the purchase or lease of its HP Indigo Digital Presses and HP Inkjet Web Presses.

REQUEST NO. 13:

All agreements between O’Neil and HP for the supply of support, training, maintenance and/or other professional services, or consumables by HP relating to its HP Indigo Digital Presses and/or HP Inkjet Web Presses.

REQUEST NO. 14:

All agreements (e.g. vendor contracts, sales agreements, licenses, development agreements) between O’Neil and any provider of the software used to practice the Accused Variable Data Printing Methods (e.g. Global Graphics, GMC Software Technology, Adobe, Quark, and Enfocus).

REQUEST NO. 15:

Documents sufficient to establish the commercial terms and the amounts paid to lease and/or own each HP Indigo Digital Press and/or HP Inkjet Web Press operated by O’Neil, including the purchase or lease price for each component and for the full system, and the commercial terms and amounts paid for consumables, product support, maintenance and/or other professional services, for each HP Indigo Digital Press and/or HP Inkjet Web Press operated by O’Neil.

REQUEST NO. 16:

For the period January 2009 to present, representative contracts or other documents sufficient to establish the commercial terms, procedures, requirements and protocols that O’Neil has established with its customers for running Variable Data Print Jobs and Static Data Print Jobs on its HP Indigo Digital Presses and HP Inkjet Web Presses.

REQUEST NO. 17:

For the period January 2009 to present, documents sufficient to establish the price setting, changes in pricing, price competition, pricing strategies for printing services, including for Variable Data Print Jobs and for Static Print Jobs.

REQUEST NO. 18:

For the period January 2009 to present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by O’Neil from the sale of printing services and/or printed materials supplied and/or produced using HP Variable Data Printing Presses, including documents sufficient to establish on a monthly and annual basis: the print volumes, pricing, gross revenue generated, any expenses O'Neil has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the HP Variable Data Printing Presses such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 19:

For the period January 2009 to present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by O'Neil from the sale of printing services and/or printed materials supplied and/or produced using the Accused Variable Data Printing Methods, including documents sufficient to establish on a monthly and annual basis: the print volumes, pricing, gross revenue generated, any expenses O'Neil has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the Accused Variable Data Printing Methods such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 20:

For the period January 2009 to present, documents sufficient to establish, on a monthly and annual basis, the percentage of print volume on the HP Variable Data Printing Presses generated using the Accused Variable Data Printing Methods, the number of Variable Data Print Jobs run, the print volume of Variable Data Print Jobs, and the customers who ordered the Variable Data Print Jobs. This request seeks available summary documents kept in the ordinary course of business such as sales reports, scheduling reports, job logs, etc. This request should not be interpreted to seek records evidencing individual transactions, such as purchase orders and invoices, unless summary documents are not available.

REQUEST NO. 21:

For the period January 2009 to present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by O'Neil from the sale of printing services and/or printed materials supplied and/or produced using HP Inkjet Web Presses, including documents sufficient to establish on a monthly and annual basis: the printing volumes, pricing, gross revenue generated, any expenses O'Neil has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the HP Inkjet Web Presses such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets.

This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 22:

For the period January 2009 to present, historical financial statements including income statements, balance sheets, statements of cash flow, and profit and loss statements for O'Neil.

REQUEST NO. 23:

Documents sufficient to identify the methodology or accounting policies used by O'Neil in calculating expenses and costs associated with its commercial printing services, such as company accounting policies, guidelines or handbooks.

REQUEST NO. 24:

Documents relating to any technology that O'Neil contends is an acceptable non-infringing alternative to the inventions claimed in the Patents-In-Suit, including documents sufficient to describe the alternative technology, to establish its availability and how it could have been implemented, and establishing the costs of using the alternative technology.

REQUEST NO. 25:

Documents referring, relating to or constituting known offers or agreements by any person or entity relating to licensing, or compensation for infringement of patents or technology comparable to any of the Patents-In-Suit, including technology relating to processing Variable Data Print Jobs or controlling InkJet print engines, including documents sufficient to calculate the royalty rate, unit volume, and amounts paid or payable for each such offer or agreement.

REQUEST NO. 26:

Documents relating to O'Neil's patent licensing practices, procedures, and policies.

REQUEST NO. 27:

All documents relating or referring to Forrest Gauthier, Varis Corporation, Tesseron, or to any of their products or systems for processing Variable Data Print Jobs.

REQUEST NO. 28:

All documents referring or relating first instance in which O'Neil first became aware of the Patents-In-Suit or any parent, continuation, divisional or continuation-in-part thereof.

REQUEST NO. 29:

Documents relating to any attempts by O'Neil, HP or their suppliers or other agents, to design around, or otherwise avoid infringement of the Patents-In-Suit.

REQUEST NO. 30:

Any and all written opinions of counsel relating to the Patents-In-Suit on which O'Neil relies in support of any defense in this case, including all information considered by counsel in rendering such opinions and all communications between O'Neil and counsel relating to the subject matter of such opinions.

REQUEST NO. 31:

All documents supporting, refuting or relating to any contention that O'Neil does or does not infringe the Patents-In-Suit.

REQUEST NO. 32:

All documents supporting, refuting or relating to any contention that the Patents-In-Suit are invalid or unenforceable.

REQUEST NO. 33:

All documents supporting, refuting or relating to any contention that O'Neil has license rights under any of the Patents-In-Suit.

REQUEST NO. 34:

All documents supporting, refuting or relating to any contention of O'Neil regarding damages in this case.

REQUEST NO. 35:

All documents supporting, refuting or relating to any defense that O'Neil intends to assert in this case, including at any hearing or trial relating to this matter.

REQUEST NO. 36:

All documents referring, relating to, or constituting any indemnity or defense agreement regarding any of the Patents-In-Suit.

CERTIFICATE OF SERVICE

I hereby certify that on August 28, 2015 a true and correct copy of the foregoing document was served by email on the recipients below:

Andrew Perito
Edward R. Reines
WEIL GOTSHAL & MANGES LLP – REDWOOD SHORES
201 Redwood Shores Parkway
5th Floor
Redwood Shores, CA 94065
Phone: (650) 802-3000
Facsimile: (650) 802-3100
Email: andrew.perito@weil.com
Email: edward.reines@weil.com

Audrey L. Maness
WEIL GOTSHAL & MANGES LLP – HOUSTON
700 Louisiana, Suite 1700
Houston, TX 77002
Phone: (713) 546-5317
Facsimile: (713) 224-9511
Email: audrey.maness@weil.com

Counsel for Defendant Hewlett Packard Co., O'Neil Data Systems, Inc., Quad/Graphics, Inc., Cenveo, Inc., Fort Dearborn Co. and Vistaprint U.S.A., Inc.

s/ Alison Aubry Richards
Alison Aubry Richards
Attorney for Plaintiff Industrial Print Technologies, LLC

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

INDUSTRIAL PRINT TECHNOLOGIES, LLC, PLAINTIFF V.	The Honorable Barbara M.G. Lynn
CENVEO, INC. AND HEWLETT-PACKARD COMPANY DEFENDANTS.	Civil Action No. 3:15-cv-00165-M

IPT'S FIRST SET OF REQUESTS FOR PRODUCTION TO DEFENDANT CENVEO, INC. ("CENVEO") (NOS. 1-35)

Pursuant to Rule 34 of the Federal Rules of Civil Procedure, Plaintiff Industrial Print Technologies LLC ("IPT") hereby requests that Defendant Cenveo, Inc. ("Cenveo") produce the documents and things requested herein for inspection and copying, to the extent not already produced, within 30 days of service hereof at the offices of Fitch, Even, Tabin & Flannery LLP, 120 South LaSalle Street, Suite 1600, Chicago, Illinois 60603, or at such other time and place agreed to by counsel.

DEFINITIONS

As used herein, the term(s):

1. “IPT” means Plaintiff Industrial Print Technologies LLC.

2. “HP” means Hewlett-Packard Company, all present and former officers, directors, employees, consultants, or other persons or entities acting in concert with them or acting on behalf of them or who are subject to the direction or control of the foregoing.

3. “Variable Data Print Job” means a printing process in which elements on a printed page may change between one instance of the page and another instance of the page in the job.

4. “Static Print Job” means a printing process in which the elements on a printed page do not change between one instance of the page and another instance of the page in the job.

5. “HP Variable Data Printing Presses” includes any printing press that is capable of processing Variable Data Print Jobs and that is manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP, including without limitation HP Indigo Digital Presses and HP Inkjet Web Presses enabled for Variable Data Print Jobs.

6. “HP Indigo Digital Presses” includes all models within the HP Indigo product line manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP (e.g., model W3250, 3550, WS4600, 5000, 5600, WS6600, WS6600p, W7250, 7500, 7600, 10000, 20000, and 30000 presses) and all components and subsystems thereof (e.g., paper supply, utility cabinet, printing engine, press computer, touch screen panel, stacker, etc.) and specifically includes each digital front end (“DFE”) associated with each such press.

7. “DFE” means digital front end, and specifically includes the raster image processor(s) (“RIP”) associated with each such press.

8. “HP Inkjet Web Presses” includes all models within the IP Inkjet Web Press product line manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP that contain multiple inkjet printheads (e.g., HP T200, T300, T350 and T400 inkjet web presses) and all components and subsystems thereof (e.g., print engine, ink delivery systems, service stations, in-line process monitor, press interface adaptor/frame broker, press controller, dryer, paper supply and rewind systems, etc.) and specifically includes each digital front end (“DFE”) associated with each such press.

9. The “Accused Variable Data Printing Methods” means the methods performed to design and create Variable Data Print Jobs for HP Variable Data Printing Presses and to process for printing and print Variable Data Print Jobs using HP Variable Data Printing Presses.

10. “Variable Data Printing Patents” means U.S. Patents Nos. 5,729,665 (“the ‘665 patent”); 5,937,153 (“the ‘153 patent”); 7,274,479 (“the ‘479 patent”); 7,333,233 (“the ‘233 patent”); and 6,381,028 (“the ’028 patent”).

11. “Patents-in-Suit” means any of the Variable Data Printing Patents individually and collectively.

12. “Programming Information” includes programs expressed in a hardware definition language (e.g. VHDL or Verilog or others) as well as software programs written for storage or execution in a hardware component. “Programming Information” includes documents describing the development software and hardware system in which the program was created and documents identifying the manner in which the program is used within the component including the precise steps that must be taken to install the program in the component.

13. “Global Graphics” means Global Graphics Software Inc. and/or Global Graphics Software Ltd. and their executives, officers, and present or former employees, as well as related companies.

14. “Printer Defendants” means Cenveo, Fort Dearborn, O’Neil, Quad Graphics, and Vistaprint.

15. “Cenveo” means Defendant Cenveo, Inc. and its executives, officers, and present or former employees, as well as related companies.

16. “Fort Dearborn” means Defendant Fort Dearborn Company and its executives, officers, and present or former employees, as well as related companies.

17. “O’Neil” means Defendant O’Neil Data Systems, Inc. and its executives, officers, and present or former employees, as well as related companies, including O’Neil Digital Solutions, LLC and O’Neil Data Systems LLC.

18. “Quad Graphics” means Defendant Quad/Graphics, Inc. and its executives, officers, and present or former employees, as well as related companies.

19. “Vistaprint” means Defendant Vistaprint USA, Inc., and its executives, officers, and present or former employees, as well as related companies, specifically including, but not limited to Vistaprint North American Services Corp., Cimpress U.S.A., Inc. and Cimpress Windsor Corporation.

INSTRUCTIONS

1. Continuing Obligation. With respect to each of the requests for production, unless otherwise expressly stated, the information sought is that which is current to the date of your answer. These requests for production are of a continuing nature. You are required to serve supplemental responses under the applicable rules if your knowledge changes in the future. Please take notice that objection will be interposed at trial to the introduction by Defendant of any evidence requested by these requests for production but not fully disclosed in your responses, including required supplementation.

2. If all or any part of a document is in a language other than English, identify and produce any translation of the non-English language document.

REQUESTS FOR DOCUMENTS AND THINGS

REQUEST NO. 1:

Documents sufficient to identify each specific HP Indigo Digital Press and/or HP Inkjet Web Press operated by Cenveo, and the physical arrangement and functional interrelations of all hardware and software components and subsystems of each HP Indigo Digital Presses and each HP Inkjet Web Press.

REQUEST NO. 2:

Documents sufficient to confirm Cenveo's operating procedures and customary practices for receiving and processing Variable Data Print Jobs and Static Data Print Jobs on its HP Indigo Digital Presses and HP Inkjet Web Presses, including documents establishing the complete work flow, from the creation and/or receipt by Cenveo of the page description specification for a print job through the fulfillment of the print job.

REQUEST NO. 3:

For each HP Variable Data Printing Press, documents sufficient to identify and describe the specific software applications used to practice the Accused Variable Data Printing Methods (e.g., Job Consumer, Indigo Press Controller, Global Graphics Harlequin), and how each software application is used.

REQUEST NO. 4:

For each software application capable of practicing the Accused Variable Data Printing Methods, documents (including software design documents, flow charts, product specifications, technical descriptions, protocol specifications, file specifications, format specifications, workflow diagrams, manuals, training information, etc.) sufficient to describe:

a. how the software identifies a static data area (i.e., an area for image elements that do not change from one instance of a page to another instance of the page) and a variable data area (i.e., an area for image elements that may change from one instance of a page to another instance of the page) of a print job described by a page description language (e.g., PDF, PPML, PPMLT, JLYT files, etc.);

b. how the software associates graphics state information (e.g., font characteristics, type size, text alignment, word wrapping, page registration parameters, fill color, angle, scale factor, etc.) with static and variable data areas;

c. how the software creates and stores bitmaps of static and variable data areas; and

d. how the software merges static data bitmaps and variable data bitmaps.

REQUEST NO. 5:

Documents identifying or discussing any standards for creation or processing of Variable Data Print Jobs that Cenveo and/or its customers use in connection with the Accused Variable Data Printing Methods, including for example, PDF/VT, PDF/X-4, PDF/X-5, PPML, PPMLT, or JLYT.

REQUEST NO. 6:

To the extent not produced in response to other Requests, documents, including reports, manuals, drawings, brochures, texts, product specifications, design specifications and source code, sufficient to establish the design, structure, operation, features and/or functions relevant to Cenveo's use of the Accused Variable Data Printing Methods on the HP Variable Data Printing Presses.

REQUEST NO. 7:

Documents describing the services (e.g., technical support, training, installation, customization, integration, and/or maintenance) that any provider of software used on the DFEs of the HP Variable Data Printing Presses has provided to Cenveo relating to the Accused Variable Data Printing Methods.

REQUEST NO. 8:

Documents referring or relating to the advantages and/or value associated with the Accused Variable Data Printing Methods (e.g., improved printing speed, increased productivity, additional throughput, reduced cost, increased printing profits, etc.) relative to alternative approaches for running Variable Data Print Jobs.

REQUEST NO. 9:

Documents such as business plans, strategic plans, business strategy reports or marketing plans discussing the strategic importance, value added, target customer market, use cases, applications, and/or market potential of Cenveo providing Variable Data Print Job printing services to the marketplace.

REQUEST NO. 10:

Documents referring to, relating to or comprising estimates or forecasts of the demand for Variable Data Print Jobs printing services in the marketplace and/or Cenveo's competition for supplying such services, including any internal estimates, estimates in trade or other press, estimates by industry analysts or the like.

REQUEST NO.11:

All agreements between Cenveo and HP for the purchase or lease of its HP Indigo Digital Presses and HP Inkjet Web Presses.

REQUEST NO. 12:

All agreements between Cenveo and HP for the supply of support, training, maintenance and/or other professional services, or consumables by HP relating to its HP Indigo Digital Presses and/or HP Inkjet Web Presses.

REQUEST NO. 13:

All agreements (e.g. vendor contracts, sales agreements, licenses, development agreements) between Cenveo and any provider of the software used to practice the Accused Variable Data Printing Methods (e.g. Global Graphics, GMC Software Technology, Adobe, Quark, and Enfocus).

REQUEST NO. 14:

Documents sufficient to establish the commercial terms and the amounts paid to lease and/or own each HP Indigo Digital Press and/or HP Inkjet Web Press operated by Cenveo, including the purchase or lease price for each component and for the full system, and the commercial terms and amounts paid for consumables, product support, maintenance and/or other professional services, for each HP Indigo Digital Press and/or HP Inkjet Web Press operated by Cenveo.

REQUEST NO. 15:

For the period January 2009 to present, representative contracts or other documents sufficient to establish the commercial terms, procedures, requirements and protocols that Cenveo has established with its customers for running Variable Data Print Jobs and Static Data Print Jobs on its HP Indigo Digital Presses and HP Inkjet Web Presses.

REQUEST NO. 16:

For the period January 2009 to present, documents sufficient to establish the price setting, changes in pricing, price competition, pricing strategies for printing services, including for Variable Data Print Jobs and for Static Print Jobs.

REQUEST NO. 17:

For the period January 2009 to the present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by Cenveo from the sale of printing services and/or printed materials supplied and/or produced using HP Variable Data Printing Presses, including documents sufficient to establish on a monthly and annual basis: the print volumes, pricing, gross revenue generated, any expenses Cenveo has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the HP Variable Data Printing Presses such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 18:

For the period January 2009 to the present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by Cenveo from the sale of printing services and/or printed materials supplied and/or produced using the Accused Variable Data Printing Methods, including documents sufficient to establish on a monthly and annual basis: the print volumes, pricing, gross revenue generated, any expenses Cenveo has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the Accused Variable Data Printing Methods such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 19:

For the period January 2009 to the present, documents sufficient to establish, on a monthly and annual basis, the percentage of print volume on the HP Variable Data Printing Presses generated using the Accused Variable Data Printing Methods, the number of Variable Data Print Jobs run, the print volume of Variable Data Print Jobs, and the customers who ordered the Variable Data Print Jobs. This request seeks available summary documents kept in the ordinary course of business such as sales reports, scheduling reports, job logs, etc. This request should not be interpreted to seek records evidencing individual transactions, such as purchase orders and invoices, unless summary documents are not available.

REQUEST NO. 20:

For the period January 2009 to the present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by Cenveo from the sale of printing services and/or printed materials supplied and/or produced using HP Inkjet Web Presses, including documents sufficient to establish on a monthly and annual basis: the printing volumes, pricing, gross revenue generated, any expenses Cenveo has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the HP Inkjet Web Presses such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 21:

For the period January 2009 to the present, historical financial statements including income statements, balance sheets, statements of cash flow, and profit and loss statements for Cenveo.

REQUEST NO. 22:

Documents sufficient to identify the methodology or accounting policies used by Cenveo in calculating expenses and costs associated with its commercial printing services, such as company accounting policies, guidelines or handbooks.

REQUEST NO. 23:

Documents relating to any technology that Cenveo contends is an acceptable non-infringing alternative to the inventions claimed in the Patents-In-Suit, including documents sufficient to describe the alternative technology, to establish its availability and how it could have been implemented, and establishing the costs of using the alternative technology.

REQUEST NO. 24:

Documents referring, relating to or constituting known offers or agreements by any person or entity relating to licensing, or compensation for infringement of patents or technology comparable to any of the Patents-In-Suit, including technology relating to processing Variable Data Print Jobs, including documents sufficient to calculate the royalty rate, unit volume, and amounts paid or payable for each such offer or agreement.

REQUEST NO. 25:

Documents relating to Cenveo’s patent licensing practices, procedures, and policies.

REQUEST NO. 26:

All documents relating or referring to Forrest Gauthier, Varis Corporation, Tesseron, or to any of their products or systems for processing Variable Data Print Jobs.

REQUEST NO. 27:

All documents referring or relating first instance in which Cenveo first became aware of the Patents-In-Suit or any parent, continuation, divisional or continuation-in-part thereof.

REQUEST NO. 28:

Documents relating to any attempts by Cenveo, HP or their suppliers or other agents, to design around, or otherwise avoid infringement of the Patents-In-Suit.

REQUEST NO. 29:

Any and all written opinions of counsel relating to the Patents-In-Suit on which Cenveo relies in support of any defense in this case, including all information considered by counsel in rendering such opinions and all communications between Cenveo and counsel relating to the subject matter of such opinions.

REQUEST NO. 30:

All documents supporting, refuting or relating to any contention that Cenveo does or does not infringe the Patents-In-Suit.

REQUEST NO. 31:

All documents supporting, refuting or relating to any contention that the Patents-In-Suit are invalid or unenforceable.

REQUEST NO. 32:

All documents supporting, refuting or relating to any contention that Cenveo has license rights under any of the Patents-In-Suit.

REQUEST NO. 33:

All documents supporting, refuting or relating to any contention of Cenveo regarding damages in this case.

REQUEST NO. 34:

All documents supporting, refuting or relating to any defense that Cenveo intends to assert in this case, including at any hearing or trial relating to this matter.

REQUEST NO. 35:

All documents referring, relating to, or constituting any indemnity or defense agreement regarding any of the Patents-In-Suit.

CERTIFICATE OF SERVICE

I hereby certify that on August 28, 2015 a true and correct copy of the foregoing document was served by email on the recipients below:

Andrew Perito
Edward R. Reines
WEIL GOTSHAL & MANGES LLP – REDWOOD SHORES
201 Redwood Shores Parkway
5th Floor
Redwood Shores, CA 94065
Phone: (650) 802-3000
Facsimile: (650) 802-3100
Email: andrew.perito@weil.com
Email: edward.reines@weil.com

Audrey L. Maness
WEIL GOTSHAL & MANGES LLP – HOUSTON
700 Louisiana, Suite 1700
Houston, TX 77002
Phone: (713) 546-5317
Facsimile: (713) 224-9511
Email: audrey.maness@weil.com

Counsel for Defendant Hewlett Packard Co., O'Neil Data Systems, Inc., Quad/Graphics, Inc., Cenveo, Inc., Fort Dearborn Co. and Vistaprint U.S.A., Inc.

s/ Alison Aubry Richards
Alison Aubry Richards
Attorney for Plaintiff Industrial Print Technologies, LLC

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF TEXAS
DALLAS DIVISION

INDUSTRIAL PRINT TECHNOLOGIES, LLC, PLAINTIFF V.	The Honorable Barbara M.G. Lynn
FORT DEARBORN COMPANY AND HEWLETT-PACKARD COMPANY DEFENDANTS.	Civil Action No. 3:15-cv-01195-M

IPT'S FIRST SET OF REQUESTS FOR PRODUCTION TO DEFENDANT FORT DEARBORN COMPANY ("FORT DEARBORN") (NOS. 1-35)

Pursuant to Rule 34 of the Federal Rules of Civil Procedure, Plaintiff Industrial Print Technologies LLC ("IPT") hereby requests that Defendant Fort Dearborn Company ("Fort Dearborn") produce the documents and things requested herein for inspection and copying, to the extent not already produced, within 30 days of service hereof at the offices of Fitch, Even, Tabin & Flannery LLP, 120 South LaSalle Street, Suite 1600, Chicago, Illinois 60603, or at such other time and place agreed to by counsel.

DEFINITIONS

As used herein, the term(s):

1. "IPT" means Plaintiff Industrial Print Technologies LLC.

2. "HP" means Hewlett-Packard Company, all present and former officers, directors, employees, consultants, or other persons or entities acting in concert with them or acting on behalf of them or who are subject to the direction or control of the foregoing.

3. "Variable Data Print Job" means a printing process in which elements on a printed page may change between one instance of the page and another instance of the page in the job.

4. "Static Print Job" means a printing process in which the elements on a printed page do not change between one instance of the page and another instance of the page in the job.

5. "HP Variable Data Printing Presses" includes any printing press that is capable of processing Variable Data Print Jobs and that is manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP, including without limitation HP Indigo Digital Presses and HP Inkjet Web Presses enabled for Variable Data Print Jobs.

6. "HP Indigo Digital Presses" includes all models within the HP Indigo product line manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP (e.g., model W3250, 3550, WS4600, 5000, 5600, WS6600, WS6600p, W7250, 7500, 7600, 10000, 20000, and 30000 presses) and all components and subsystems thereof (e.g., paper supply, utility cabinet, printing engine, press computer, touch screen panel, stacker, etc.) and specifically includes each digital front end ("DFE") associated with each such press.

7. “DFE” means digital front end, and specifically includes the raster image processor(s) (“RIP”) associated with each such press.

8. “HP Inkjet Web Presses” includes all models within the IP Inkjet Web Press product line manufactured, sold, offered for sale, imported, supplied, and/or operated by HP or any entity under the control or direction of HP that contain multiple inkjet printheads (e.g., HP T200, T300, T350 and T400 inkjet web presses) and all components and subsystems thereof (e.g., print engine, ink delivery systems, service stations, in-line process monitor, press interface adaptor/frame broker, press controller, dryer, paper supply and rewind systems, etc.) and specifically includes each digital front end (“DFE”) associated with each such press.

9. The “Accused Variable Data Printing Methods” means the methods performed to design and create Variable Data Print Jobs for HP Variable Data Printing Presses and to process for printing and print Variable Data Print Jobs using HP Variable Data Printing Presses.

10. “Variable Data Printing Patents” means U.S. Patents Nos. 5,729,665 (“the ‘665 patent”); 5,937,153 (“the ‘153 patent”); 7,274,479 (“the ‘479 patent”); 7,333,233 (“the ‘233 patent”); and 6,381,028 (“the ’028 patent”).

11. “Patents-in-Suit” means any of the Variable Data Printing Patents individually and collectively.

12. “Programming Information” includes programs expressed in a hardware definition language (e.g. VHDL or Verilog or others) as well as software programs written for storage or execution in a hardware component. “Programming Information” includes documents describing the development software and hardware system in which the program was created and documents identifying the manner in which the program is used within the component including the precise steps that must be taken to install the program in the component.

13. “Global Graphics” means Global Graphics Software Inc. and/or Global Graphics Software Ltd. and their executives, officers, and present or former employees, as well as related companies.

14. “Printer Defendants” means Cenveo, Fort Dearborn, O’Neil, Quad Graphics, and Vistaprint.

15. “Cenveo” means Defendant Cenveo, Inc. and its executives, officers, and present or former employees, as well as related companies.

16. “Fort Dearborn” means Defendant Fort Dearborn Company and its executives, officers, and present or former employees, as well as related companies.

17. “O’Neil” means Defendant O’Neil Data Systems, Inc. and its executives, officers, and present or former employees, as well as related companies, including O’Neil Digital Solutions, LLC and O’Neil Data Systems LLC.

18. “Quad Graphics” means Defendant Quad/Graphics, Inc. and its executives, officers, and present or former employees, as well as related companies.

19. “Vistaprint” means Defendant Vistaprint USA, Inc., and its executives, officers, and present or former employees, as well as related companies, specifically including, but not limited to Vistaprint North American Services Corp., Cimpress U.S.A., Inc. and Cimpress Windsor Corporation.

INSTRUCTIONS

1. Continuing Obligation. With respect to each of the requests for production, unless otherwise expressly stated, the information sought is that which is current to the date of your answer. These requests for production are of a continuing nature. You are required to serve supplemental responses under the applicable rules if your knowledge changes in the future. Please take notice that objection will be interposed at trial to the introduction by Defendant of any evidence requested by these requests for production but not fully disclosed in your responses, including required supplementation.

2. If all or any part of a document is in a language other than English, identify and produce any translation of the non-English language document.

REQUESTS FOR DOCUMENTS AND THINGS

REQUEST NO. 1:

Documents sufficient to identify each specific HP Indigo Digital Press and/or HP Inkjet Web Press operated by Fort Dearborn, and the physical arrangement and functional interrelations of all hardware and software components and subsystems of each HP Indigo Digital Presses and each HP Inkjet Web Press.

REQUEST NO. 2:

Documents sufficient to confirm Fort Dearborn's operating procedures and customary practices for receiving and processing Variable Data Print Jobs and Static Data Print Jobs on its HP Indigo Digital Presses and HP Inkjet Web Presses, including documents establishing the complete work flow, from the creation and/or receipt by Fort Dearborn of the page description specification for a print job through the fulfillment of the print job.

REQUEST NO. 3:

For each HP Variable Data Printing Press, documents sufficient to identify and describe the specific software applications used to practice the Accused Variable Data Printing Methods (e.g., Job Consumer, Indigo Press Controller, Global Graphics Harlequin), and how each software application is used.

REQUEST NO. 4:

For each software application capable of practicing the Accused Variable Data Printing Methods, documents (including software design documents, flow charts, product specifications, technical descriptions, protocol specifications, file specifications, format specifications, workflow diagrams, manuals, training information, etc.) sufficient to describe:

a. how the software identifies a static data area (i.e., an area for image elements that do not change from one instance of a page to another instance of the page) and a variable data area (i.e., an area for image elements that may change from one instance of a page to another instance of the page) of a print job described by a page description language (e.g., PDF, PPML, PPMLT, JLYT files, etc.);

b. how the software associates graphics state information (e.g., font characteristics, type size, text alignment, word wrapping, page registration parameters, fill color, angle, scale factor, etc.) with static and variable data areas;

c. how the software creates and stores bitmaps of static and variable data areas; and

d. how the software merges static data bitmaps and variable data bitmaps.

REQUEST NO. 5:

Documents identifying or discussing any standards for creation or processing of Variable Data Print Jobs that Fort Dearborn and/or its customers use in connection with the Accused Variable Data Printing Methods, including for example, PDF/VT, PDF/X-4, PDF/X-5, PPML, PPMLT, or JLYT.

REQUEST NO. 6:

To the extent not produced in response to other Requests, documents, including reports, manuals, drawings, brochures, texts, product specifications, design specifications and source code, sufficient to establish the design, structure, operation, features and/or functions relevant to Fort Dearborn's use of the Accused Variable Data Printing Methods on the HP Variable Data Printing Presses.

REQUEST NO. 7:

Documents describing the services (e.g., technical support, training, installation, customization, integration, and/or maintenance) that any provider of software used on the DFEs of the HP Variable Data Printing Presses has provided to Fort Dearborn relating to the Accused Variable Data Printing Methods.

REQUEST NO. 8:

Documents referring or relating to the advantages and/or value associated with the Accused Variable Data Printing Methods (e.g., improved printing speed, increased productivity, additional throughput, reduced cost, increased printing profits, etc.) relative to alternative approaches for running Variable Data Print Jobs.

REQUEST NO. 9:

Documents such as business plans, strategic plans, business strategy reports or marketing plans discussing the strategic importance, value added, target customer market, use cases, applications, and/or market potential of Fort Dearborn providing Variable Data Print Job printing services to the marketplace.

REQUEST NO. 10:

Documents referring to, relating to or comprising estimates or forecasts of the demand for Variable Data Print Jobs printing services in the marketplace and/or Fort Dearborn's competition for supplying such services, including any internal estimates, estimates in trade or other press, estimates by industry analysts or the like.

REQUEST NO.11:

All agreements between Fort Dearborn and HP for the purchase or lease of its HP Indigo Digital Presses and HP Inkjet Web Presses.

REQUEST NO. 12:

All agreements between Fort Dearborn and HP for the supply of support, training, maintenance and/or other professional services, or consumables by HP relating to its HP Indigo Digital Presses and/or HP Inkjet Web Presses.

REQUEST NO. 13:

All agreements (e.g. vendor contracts, sales agreements, licenses, development agreements) between Fort Dearborn and any provider of the software used to practice the Accused Variable Data Printing Methods (e.g. Global Graphics, GMC Software Technology, Adobe, Quark, and Enfocus).

REQUEST NO. 14:

Documents sufficient to establish the commercial terms and the amounts paid to lease and/or own each HP Indigo Digital Press and/or HP Inkjet Web Press operated by Fort Dearborn, including the purchase or lease price for each component and for the full system, and the commercial terms and amounts paid for consumables, product support, maintenance and/or other professional services, for each HP Indigo Digital Press and/or HP Inkjet Web Press operated by Fort Dearborn.

REQUEST NO. 15:

For the period January 2009 to present, representative contracts or other documents sufficient to establish the commercial terms, procedures, requirements and protocols that Fort Dearborn has established with its customers for running Variable Data Print Jobs and Static Data Print Jobs on its HP Indigo Digital Presses and HP Inkjet Web Presses.

REQUEST NO. 16:

For the period January 2009 to present, documents sufficient to establish the price setting, changes in pricing, price competition, pricing strategies for printing services, including for Variable Data Print Jobs and for Static Print Jobs.

REQUEST NO. 17:

For the period January 2009 to the present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by Fort Dearborn from the sale of printing services and/or printed materials supplied and/or produced using HP Variable Data Printing Presses, including documents sufficient to establish on a monthly and annual basis: the print volumes, pricing, gross revenue generated, any expenses Fort Dearborn has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the HP Variable Data Printing Presses such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 18:

For the period January 2009 to the present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by Fort Dearborn from the sale of printing services and/or printed materials supplied and/or produced using the Accused Variable Data Printing Methods, including documents sufficient to establish on a monthly and annual basis: the print volumes, pricing, gross revenue generated, any expenses Fort Dearborn has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the Accused Variable Data Printing Methods such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 19:

For the period January 2009 to the present, documents sufficient to establish, on a monthly and annual basis, the percentage of print volume on the HP Variable Data Printing Presses generated using the Accused Variable Data Printing Methods, the number of Variable Data Print Jobs run, the print volume of Variable Data Print Jobs, and the customers who ordered the Variable Data Print Jobs. This request seeks available summary documents kept in the ordinary course of business such as sales reports, scheduling reports, job logs, etc. This request should not be interpreted to seek records evidencing individual transactions, such as purchase orders and invoices, unless summary documents are not available.

REQUEST NO. 20:

For the period January 2009 to the present, documents sufficient to establish monthly and annual revenues, expenses, gross profit and net profit generated by Fort Dearborn from the sale of printing services and/or printed materials supplied and/or produced using HP Inkjet Web Presses, including documents sufficient to establish on a monthly and annual basis: the printing volumes, pricing, gross revenue generated, any expenses Fort Dearborn has deducted to calculate its gross and net profit, and the resulting gross and net profit. This request seeks available summary documents kept in the ordinary course of business which calculate, set forth, or report the print volumes, revenues, expenses, gross profit, net profit associated with printing services and/or printed materials associated with the HP Inkjet Web Presses such as monthly, quarterly or annual sales reports, profit and loss statements, income sheets, statements of cash flow, or balance sheets. This request should not be interpreted to seek records evidencing individual transactions, such as invoices, unless summary documents are not available.

REQUEST NO. 21:

For the period January 2009 to the present, historical financial statements including income statements, balance sheets, statements of cash flow, and profit and loss statements for Fort Dearborn.

REQUEST NO. 22:

Documents sufficient to identify the methodology or accounting policies used by Fort Dearborn in calculating expenses and costs associated with its commercial printing services, such as company accounting policies, guidelines or handbooks.

REQUEST NO. 23:

Documents relating to any technology that Fort Dearborn contends is an acceptable non-infringing alternative to the inventions claimed in the Patents-In-Suit, including documents sufficient to describe the alternative technology, to establish its availability and how it could have been implemented, and establishing the costs of using the alternative technology.

REQUEST NO. 24:

Documents referring, relating to or constituting known offers or agreements by any person or entity relating to licensing, or compensation for infringement of patents or technology comparable to any of the Patents-In-Suit, including technology relating to processing Variable Data Print Jobs, including documents sufficient to calculate the royalty rate, unit volume, and amounts paid or payable for each such offer or agreement.

REQUEST NO. 25:

Documents relating to Fort Dearborn's patent licensing practices, procedures, and policies.

REQUEST NO. 26:

All documents relating or referring to Forrest Gauthier, Varis Corporation, Tesseron, or to any of their products or systems for processing Variable Data Print Jobs.

REQUEST NO. 27:

All documents referring or relating first instance in which Fort Dearborn first became aware of the Patents-In-Suit or any parent, continuation, divisional or continuation-in-part thereof.

REQUEST NO. 28:

Documents relating to any attempts by Fort Dearborn, HP or their suppliers or other agents, to design around, or otherwise avoid infringement of the Patents-In-Suit.

REQUEST NO. 29:

Any and all written opinions of counsel relating to the Patents-In-Suit on which Fort Dearborn relies in support of any defense in this case, including all information considered by counsel in rendering such opinions and all communications between Fort Dearborn and counsel relating to the subject matter of such opinions.

REQUEST NO. 30:

All documents supporting, refuting or relating to any contention that Fort Dearborn does or does not infringe the Patents-In-Suit.

REQUEST NO. 31:

All documents supporting, refuting or relating to any contention that the Patents-In-Suit are invalid or unenforceable.

REQUEST NO. 32:

All documents supporting, refuting or relating to any contention that Fort Dearborn has license rights under any of the Patents-In-Suit.

REQUEST NO. 33:

All documents supporting, refuting or relating to any contention of Fort Dearborn regarding damages in this case.

REQUEST NO. 34:

All documents supporting, refuting or relating to any defense that Fort Dearborn intends to assert in this case, including at any hearing or trial relating to this matter.

REQUEST NO. 35:

All documents referring, relating to, or constituting any indemnity or defense agreement regarding any of the Patents-In-Suit.

CERTIFICATE OF SERVICE

I hereby certify that on August 28, 2015 a true and correct copy of the foregoing document was served by email on the recipients below:

Andrew Perito
Edward R. Reines
WEIL GOTSHAL & MANGES LLP – REDWOOD SHORES
201 Redwood Shores Parkway
5th Floor
Redwood Shores, CA 94065
Phone: (650) 802-3000
Facsimile: (650) 802-3100
Email: andrew.perito@weil.com
Email: edward.reines@weil.com

Audrey L. Maness
WEIL GOTSHAL & MANGES LLP – HOUSTON
700 Louisiana, Suite 1700
Houston, TX 77002
Phone: (713) 546-5317
Facsimile: (713) 224-9511
Email: audrey.maness@weil.com

Counsel for Defendant Hewlett Packard Co., O'Neil Data Systems, Inc., Quad/Graphics, Inc., Cenveo, Inc., Fort Dearborn Co. and Vistaprint U.S.A., Inc.

s/ Alison Aubry Richards
Alison Aubry Richards
Attorney for Plaintiff Industrial Print Technologies, LLC

DOCUMENTS PRODUCED BY HP BETWEEN JUNE 7 AND JUNE 24, 2016

Date	Production Numbers	Number of Documents	Number of Pages	General Overview of Contents
6/7/2016	IPTHP00025938–26120	9	183	Documents regarding HP’s SmartStream Production Pro product (e.g., brochures, data sheets, presentations).
6/21/2016	IPTHP00342617–363157	1,749	20,541	Documents regarding Label & Packaging press products and solutions, SmartStream, Esko DFE development (e.g., presentations, use cases, weekly meeting minutes, draft user guides, brochures, requirements), etc.
6/23/2016	IPTHP00026121–26761	53	641	Documents regarding HP’s SmartStream Designer product (e.g., brochures, presentations, customer-facing materials, draft licenses).
6/23/2016	IPTHP00026762–27284	43	523	Contracts, agreements with HP Indigo customers.
6/23/2016	IPTHP00363158–448191	32,785	85,034	Contracts, agreements with HP Indigo customers; garbage files (e.g., IPTHP00365165–302, 365343, 365355–57, 448076–191).
6/24/2016	IPTHP00448192–480123	5,254	31,932	Source code, user manuals (including foreign language versions), help files, status charts and meeting minutes for various DFE projects, documents regarding JLYT and PPML (e.g., IPTHP00453687–455833), Yours Truly Designer product, DFE screenshots, Project Qualification Documents (requested following the deposition of Scott Cazel in May 2016), etc.